RANCANG BANGUN APLIKASI *MOBILE GEOTAGGING* KERUSAKAN JALAN BERBASIS LAPORAN SOSIAL PADA *PLATFORM* ANDROID

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Murtadho
NIM: 115060807111153

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

Rancang Bangun Aplikasi *Mobile Geotagging* Kerusakan Jalan Berbasis Laporan Sosial pada *Platform* Android

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Murtadho
NIM: 115060807111153

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng. Herman Tolle, S.T., M.T.	Agi Putra Kharisma, S.T., M.T.
NIP. 19740823 200012 1 001	NIK: 201304 860430 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP. 19710518 2003121 1 001

IDENTITAS TIM PENGUJI

Penguji 1

Nama : Ratih Kartika Dewi, S.T., M.Kom.

NIK : 201503 890520 2 001

Penguji 2

Nama : Issa Arwani, S.Kom., M.Sc.

NIP : 19830922 201212 1 003

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 3 Agustus 2018

Akhmad Syururi
NIM: 115060807111092

DAFTAR RIWAYAT HIDUP

Nama : Muhammad Murtadho

Tempat, Tanggal Lahir : Lamongan, 29 Mei 1993

Riwayat Sekolah : MI Murni Sunan Drajat Lamongan

SMPN 1 Lamongan

SMAN 2 Lamongan

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Allah SWT Yang Maha Pengasih dan Penyayang, karena dengan berkat dan anugerah-Nya, penulis dapat menyelesaikan skripsi yang berjudul "**Rancang Bangun Aplikasi Mobile Geotagging Kerusakan Jalan Berbasis Laporan Sosial pada Platform Android**". Skripsi ini disusun untuk memenuhi sebagian persyaratan dalam memperoleh gelar Sarjana Komputer.

Penulis menyadari bahwa dengan segala kekurangan yang penulis miliki, penyusunan skripsi ini tidak dapat terselesaikan tanpa bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis mengucapkan terima kasih kepada semua yang telah bersedia untuk memberikan bantuan dan dukungan demi kelancaran penyusunan skripsi ini. Terima kasih penulis sampaikan kepada:

1. Ibu, Aba, kakak-kakak, nenek, serta seluruh keluarga penulis atas aliran doa yang tak ada henti-hentinya, motivasi, kasih sayang, juga dukungan moril dan materil sebagai penyemangat dalam menyelesaikan skripsi ini.
2. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D. selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
3. Bapak Bayu Priyambadha, S.Kom., M.Kom. selaku Ketua Program Studi Teknik Informatika Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Dr. Eng. Herman Tolle, S.T., M.T. dan Bapak Agi Putra Kharisma, S.T., M.T. selaku dosen pembimbing skripsi yang telah membimbing dan mengarahkan penulis sehingga dapat menyelesaikan skripsi ini.
5. Sahabat sekelas penulis semasa kuliah Irfandi Achmad, Adi Wiratama, Rio Trilaksono, Richa Etika, Mareta Rizky, Rohbi Visdya Harris, Heddy Sebastian, Nur Muhamad Rashid, Fikhi Nugroho, Rosikhan Maulana, E. Tito Valaddo, Dyah Pramesti, dan lainnya yang telah menemani, memberikan dukungan, contoh, dan semangat dalam pengerjaan skripsi ini.
6. Sahabat Informatika angkatan 2011 pejuang skripsi terakhir Ahmad Dzulfikar, Stefanus Bayu, Rizky Kharisma, Moh. Wahyudi, Fahmi, Arif, Aslim, Alif, Sena, dan Aidil serta rekan lainnya yang belum sempat disebutkan yang telah menemani baik di dalam kampus maupun di luar kampus, berjuang bersama dan saling membantu dalam penyelesaian skripsi.
7. Teman dekat penulis, Yuwilda Wilantikasari atas semangat, motivasi, doa, serta penghiburan yang diberikan kepada penulis dalam penyelesaian penelitian ini.
8. Sahabat-sahabat yang telah lulus mendahului saya Dimas Atmojo, Aula Rieza, Fadjrin Hidayah, Syaifuddin Zuhri, Danny Putra, Johan Prasetyo, dan lainnya, atas motivasinya untuk mengikuti langkah mereka menyelesaikan pendidikan S1 di kampus tercinta.

Hanya rasa terima kasih dan doa yang dapat penulisan berikan, semoga Allah SWT memberikan balasan yang setimpal atas jasa dan amal baik yang telah diberikan. Pada akhirnya, penulis menyadari bahwa skripsi ini masih belum sempurna. Oleh karena itu, penulis sangat mengharapkan kritik dan saran yang

membangun. Penulis berharap semoga skripsi ini dapat bermanfaat bagi pengembangan ilmu pengetahuan dan teknologi.

Malang, 3 Agustus 2018

Penulis

tadho@me.com

ABSTRAK

Muhammad Murtadho, Rancang Bangun Aplikasi *Mobile Geotagging* Kerusakan Jalan Berbasis Laporan Sosial Pada Platform Android.

Pembimbing: Dr.Eng. Herman Tolle, S.T., M.T., Agi Putra Kharisma, S.T., M.T.

Infrastruktur merupakan salah satu unsur penting dalam menggerakkan pertumbuhan ekonomi dan pembangunan. Dalam hal ini, infrastruktur jalan memegang kunci penting untuk kelangsungan aktifitas sehari-hari bagi banyak pihak di kalangan masyarakat. Ditambah dengan semakin meningkatnya jumlah pengguna kendaraan bermotor sebagai salah satu pihak pengguna infrastruktur jalan terbesar, tentunya dibutuhkan sarana infrastruktur jalan yang memadahi. Kerusakan infrastruktur jalan, terutama yang banyak dilalui oleh pengguna jalan, dapat berdampak signifikan terhadap terhambatnya aktifitas sehari-hari. Di lain sisi, perangkat bergerak juga mengalami peningkatan pengguna secara pesat dari tahun ke tahun. Di antara berbagai macam *platform* perangkat bergerak, Android merupakan *platform* dengan jumlah pengguna terbanyak dibandingkan dengan *platform* lain di Indonesia. Hampir seluruh perangkat bergerak Android dilengkapi dengan berbagai macam sensor, seperti GPS dan kamera. Salah satu implementasi penggunaan sensor GPS dan kamera pada perangkat bergerak adalah teknologi *geotagging*. Teknologi *geotagging* dapat menyematkan informasi koordinat suatu lokasi pada gambar digital. Dalam penelitian ini, penulis merancang dan membangun aplikasi bernama 'Markgo', yang memiliki fungsi untuk dapat secara mudah melaporkan adanya kerusakan jalan. Aplikasi ini ditujukan untuk masyarakat umum sebagai pengguna jalan. Setelah dilakukan pengujian dalam penelitian ini terhadap aplikasi Markgo, didapatkan tingkat validasi 100% dengan terpenuhinya 8 kebutuhan fungsional pada aplikasi. Selain itu, juga dilakukan pengujian penggunaan aplikasi yang meliputi 4 kriteria, yaitu *usefulness, ease to use, ease to learn,* dan *satisfaction*, yang mendapatkan presentase nilai indeks rata-rata sebesar 88,24%. Dari kedua hasil pengujian tersebut dapat disimpulkan bahwa 'Markgo' berfungsi sesuai dengan rancangan dan dapat diterima dan memudahkan masyarakat dalam melakukan pelaporan kerusakan jalan.

Kata kunci: *infrastruktur jalan, perangkat bergerak, GPS, LBS*

ABSTRACT

Muhammad Murtadho, Rancang Bangun Aplikasi *Mobile Geotagging* Kerusakan Jalan Berbasis Laporan Sosial Pada Platform Android.

Pembimbing: Dr.Eng. Herman Tolle, S.T., M.T., Agi Putra Kharisma, S.T., M.T.

Infrastructure is an important element in driving economic growth and public development. In this case, the road infrastructure holds an important role to the continuity of daily activities for many in the society. Along with the increasing number of vehicles users as one of the largest road users, the need of a complete road infrastructure is very important. Damaged road infrastructure, particullary the crowdy one, can significantly hampered daily activities. On the other hand, mobile devices are also experiencing a rapid increase in users from year to year. Among the various mobile device platforms, Android is the platform with the largest number of users compared to other platforms in Indonesia. Almost all Android mobile devices are equipped with various sensors, such as GPS and camera. One implementation of the GPS and cameras sensor on mobile devices is geotagging technology. Geotagging technology can embed coordinate information of a location on a digital image. In this research, the authors designed and built an application called 'Markgo'*, which has the functionality to easily report damaged road. This application is intended for the general public as road users. With the validation testing of* 'Markgo'*, the author obtained the result of 100% validity by fullfiling 8 functional requirements. In addition, from the usability testing of the application which includes 4 criteria, usefulness, ease to use, ease to learn, and satisfaction, we obtained the percentage of the average index value of 88.24%. From both test results it can be concluded that* 'Markgo' *functionality is in accordance with the design and can be accepted and facilitate the society in reporting damaged road.*

Keywords: Road infrastructure, mobile devices, GPS, location-based service

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE

BAB 1 PENDAHULUAN

1.1 Latar belakang

Teknologi *smartphone* menjadi kebutuhan yang sudah sangat erat dalam kehidupan sehari-hari. *Smartphone* sebagai media komunikasi yang memiliki berbagai macam fitur, juga memiliki keuntungan yang sangat mendukung kegunaannya, yaitu fleksibel dapat dibawa ke mana saja dan kapan saja serta memiliki '*user-friendly interface*'. Sistem operasi Android menjadi sistem operasi *smartphone* yang paling marak diminati oleh pengguna dan pengembang Android (Holzer & Ondrus, 2011). Menurut Statcounter Juni 2015, data statistik penggunaan web di Indonesia sebagian besar menggunakan perangkat Android, yakni sebesar 62,22% (Statcounter, 2015).

Pada perangkat *smartphone* sekarang ini, teknologi GPS dan kamera sudah menjadi fitur dasar. Salah satu implementasi dari teknologi GPS dan kamera yaitu *geotagging* yang merupakan teknologi untuk memberikan keterangan berbasis lokasi pada media digital yang diambil menggunakan perangkat *smartphone*. Teknologi *geotagging* banyak diimplementasikan pada jejaring sosial untuk menampilkan di mana lokasi suatu foto diambil sehingga pengguna dapat berbagi informasi secara lebih mendetail (Valli, C. 2010).

Di sisi lain, infrastruktur memegang peranan penting sebagai salah satu roda penggerak pertumbuhan ekonomi dan pembangunan. Keberadaan infrastruktur, khususnya jalan yang memadai sangat diperlukan. Jalan merupakan prasarana infrastruktur dasar yang dibutuhkan manusia untuk dapat melakukan pergerakan dari suatu lokasi ke lokasi lainnya dalam rangka pemenuhan kebutuhan. Ketersediaan jalan menjadi hal yang dianggap mendesak manakala kegiatan ekonomi masyarakat mengalami pertumbuhan yang cukup signifikan (Nagara, C. 2009).

Di Kota Malang, banyak keluhan kerusakan jalan yang dilaporkan setiap musim penghujan. Menurut kepala DPUPR Kota Malang 2018, kerusakan jalan pada umumnya disebabkan oleh genangan air hujan yang membuat aspal jalan mengelupas. Selain itu juga diakibatkan oleh beban lalu lintas yang melebihi beban rencana pembangunan. Pemerintah Kota Malang juga memiliki target untuk pembangunan jalan layak hingga mencapai 98,8%. Tetapi sampai saat ini, untuk melakukan pelaporan kerusakan jalan, masyarakat masih harus melaporkan secara langsung kepada dinas terkait (Hardiyanto, 2018). Oleh karena itu, sarana untuk memudahkan pelaporan jika terjadi gangguan kerusakan infrastruktur jalan sangat dibutuhkan.

Dari paparan informasi di atas, penulis menawarkan solusi pelaporan kerusakan jalan memanfaatkan teknologi *geotagging* pada perangkat Android *smartphone* agar dapat memudahkan masyarakat sebagai pengguna jalan. Dengan perangkat Android *smartphone* yang sudah sangat populer di kalangan masyarakat saat ini, diharapkan pelaporan kerusakan jalan dapat dilakukan oleh

hampir semua kalangan. Juga karena tingkat kepraktisan dan mobilitas *smartphone*, sehingga memungkinkan untuk melakukan pelaporan di mana saja.

Berdasarkan latar belakang tersebut penulis mengajukan judul penelitian "Rancang Bangun Aplikasi *Mobile Geotagging* Kerusakan Jalan Berbasis Laporan Sosial Pada *Platform* Android" dengan harapan aplikasi ini nantinya dapat memudahkan pengguna, khususnya masyarakat agar mendapat sarana pelaporan kerusakan jalan yang mudah dan efektif.

1.2 Rumusan masalah

Berdasarkan paparan latar belakang tersebut, maka rumusan masalah yang bisa dikaji adalah sebagai berikut:

1. Bagaimana proses analisis kebutuhan dari aplikasi *mobile geotagging* kerusakan jalan yang dapat memudahkan masyarakat untuk melaporkan kerusakan jalan?
2. Bagaimana rancangan aplikasi *mobile geotagging* kerusakan jalan yang dapat memudahkan masyarakat untuk melaporkan kerusakan jalan secara *online*?
3. Bagaimana implementasi untuk memudahkan masyarakat melihat informasi kerusakan jalan berdasarkan lokasi?
4. Bagaimana hasil pengujian validasi dan *usability* aplikasi *mobile geotagging* kerusakan jalan yang telah dirancang dan dibangun?

1.3 Tujuan

Tujuan dari penelitian ini adalah :

1. Menganalisis kebutuhan apa saja yang diperlukan untuk memudahkan masyarakat dalam melaporkan kerusakan jalan.
2. Merancang aplikasi *geotagging* kerusakan jalan yang memudahkan masyarakat untuk dapat secara *online* melaporkan kerusakan jalan.
3. Membangun sistem laporan kerusakan jalan yang dapat memudahkan masyarakat untuk melihat informasi terkait kerusakan jalan berbasis lokasi.
4. Menguji validasi dan *usability* dari implementasi aplikasi *mobile geotagging* kerusakan jalan berdasarkan rancangan yang telah dibuat sebelumnya.

1.4 Manfaat

Manfaat dari penelitian ini adalah:

1. Memudahkan masyarakat untuk melaporkan kerusakan infrastruktur jalan secara langsung.
2. Memudahkan masyarakat dan pihak pemerintah untuk melihat informasi tentang kerusakan jalan.

1.5 Batasan masalah

Batasan masalah dalam penelitian ini adalah:

1. Aplikasi pada *smartphone* memerlukan Android 5.0 ke atas dan harus terpasang Google Play Service.

2. Aplikasi hanya sebagai sarana pelaporan kerusakan jalan, belum terintegrasi dengan dinas terkait untuk menangani kerusakan jalan.
3. Aplikasi web admin dibangun menggunakan bahasa HTML, CSS, dan Javascipt, yang memiliki fungsi dasar untuk mengelola data laporan dan pengguna.
4. Cakupan wilayah aplikasi hanya sebatas Kota dan Kabupaten Malang, Jawa Timur.
5. Data yang dilaporkan berupa data foto, koordinat lokasi, dan keterangan pelapor yang bersifat opsional.

1.6 Sistematika pembahasan

Sistematika penulisan sebagai berikut:

BAB 1 : Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

BAB 2 : Tinjauan Pustaka

Pada tinjauan pustaka membahas tentang dasar-dasar teori yang mendukung rancang bangun aplikasi *mobile geotagging* kerusakan jalan berbasis laporan masyarakat pada *platform* Android.

BAB 3 : Metodologi Penelitian

Pada bab ini dijelaskan gambaran umum yang meliputi: observasi literatur, penggunaan dasar teori, analisis kebutuhan perangkat lunak, metode pengambilan data, serta pemilihan bentuk pengujian yang tepat.

BAB 4 : Analisis Kebutuhan dan Perancangan Sistem

Membahas tentang bagaimana tahapan-tahapan dari perancangan sistem aplikasi yang akan dibangun.

BAB 5 : Implementasi Program

Bab ini berisi tentang realisasi dari perancangan berdasarkan metodologi dan perancangan.

BAB 6 : Pengujian dan Analisis

Membahas tentang pengujian aplikasi pada perangkat Android *smartphone*.

BAB 7 : Penutup

Memuat kesimpulan yang diperoleh dari pembuatan, pengujian, dan saran terhadap penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini dilakukan pembahasan mengenai kajian pustaka dan teori dasar yang berkaitan dengan penelitian ini. Teori yang akan dibahas adalah laporan sosial, *geotagging*, aplikasi perangkat bergerak, bahasa pemrograman java, Google Maps API, *Javascript Object Notation*, *Web Service*, *REST*, UML, konsep dasar dari Android *Operating System* pada aplikasi *client*.

2.1 Laporan Sosial

Sistem laporan sosial merupakan sistem yang menggunakan laporan dari badan sosial sebagai sumber data laporan. Laporan sosial dapat memberikan gambaran terhadap keadaan yang terjadi beserta perkembangannya dalam suatu lingkungan yang ditentukan (Hughes, 2010). Tujuan umum dari laporan sosial antara lain:

1. Mengumpulkan data laporan dari suatu lingkungan yang ditentukan.
2. Kontribusi dalam laporan yang bersifat terbuka.
3. Membantu perencanaan dan pengambilan keputusan dari kondisi yang ada.
4. Mengidentifikasi area tujuan untuk dilakukan tindakan selanjutnya.

2.2 *Geotagging*

Geotagging adalah proses penambahan identifikasi geografis dengan cara menyematkan *metadata* pada berbagai macam media seperti foto, video, website, pesan singkat, *QR code* (Denso Wave, 2015). Data *geotagging* biasanya berupa data koordinat garis lintang dan garis bujur yang disertai dengan data sisi belahan bumi.

Geotagging memudahkan pengguna untuk mencari berbagai macam informasi spesifik mengenai lokasi melalui perangkat yang digunakan. Misalnya mencari lokasi foto terdekat dengan memasukkan data koordinat lokasi pada *image search engine* terkait. Layanan informasi *geotagging* juga dapat membantu pengguna untuk mencari berita, website, dan lain-lain berdasarkan lokasi (Jesdanun, 2012).

2.3 Aplikasi Perangkat Bergerak

Aplikasi perangkat bergerak merupakan program komputer yang didesain untuk berjalan pada *smartphone*, tablet, dan berbagai macam perangkat bergerak lain. Umumnya aplikasi perangkat bergerak didistribusikan melalui *application distribution platforms* yang dioperasikan oleh pemilik *operating system* perangkat bergerak seperti Apple App Store, Google Play, Blackberry App World, Windows Phone Store. Apple App Store mulai mempopulerkan distribusi aplikasi perangkat bergerak pada tahun 2008 (Siegler, 2012).

2.4 Bahasa Pemrograman Java

Java adalah bahasa pemrograman yang berdasar pada bahasa C++. Java dikembangkan oleh perusahaan Sun Microsystems pada Agustus 1991. Java semula bernama Oak, diambil dari nama pohon oak yang terlihat dari jendela kamar pembuatnya, James Gosling. Kemudian berganti nama menjadi Green. Pada akhirnya, pembuatnya memutuskan untuk menggunakan nama Java yang diambil dari kopi Jawa (Kabutz, 2011). Pada tahun 2009-2010, Sun Microsystems diakuisisi oleh Oracle. Dukungan Java versi terbaru, yaitu versi 8, adalah versi satu-satunya yang dikelola oleh Oracle secara gratis saat ini. Google memutuskan untuk menggunakan bahasa pemrograman Java sebagai dasar Android SDK (Rosenblatt, 2014).

2.5 Google Maps API

Google Maps API merupakan antarmuka pemrograman dari layanan Google Maps yang memungkinkan pengembang aplikasi menggunakan layanan Google Maps pada aplikasinya. Google Maps API dapat digunakan di berbagai platform, seperti web, iOS, atau Android.

Gambar 2.1 *Tile* peta Google Maps (IOActive Inc., 2011)

Google Maps menampilkan peta dengan cara membagi tampilan layar dengan luas yang telah ditentukan menjadi beberapa bagian *tile* seperti pada gambar 2.1. Ada beberapa jenis *tile*, contohnya *tile* satelit, peta, dan *overlay* yang menunjukkan wilayah negara, kota, nama jalan dengan cara memuat gambar transparan berformat PNG. Selanjutnya untuk menunjuk suatu lokasi, Google Maps menggunakan *encoding* x,y,z untuk koordinat bujur, lintang, dan tingkat zoom. Misalnya untuk menampilkan *tile* pada lokasi x=1,y=2,z=3, Google Maps API akan mengirim HTTP *request* menuju URL : ...google.com/blablabla?x=1&y=2&z=3 (IOActive Inc., 2011).

2.6 Javascript *Object Notation* (JSON)

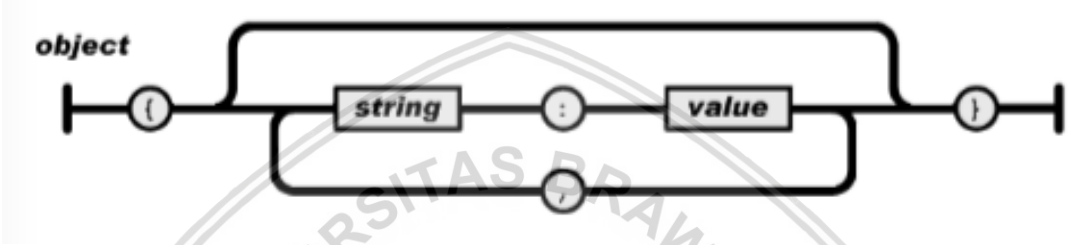
Javascript *Object Notation* (JSON) merupakan notasi format standar terbuka yang sering digunakan untuk pertukaran data antara sisi, misal *client-server*. Data JSON biasanya berupa format teks yang dapat dipahami manusia, menggunakan struktur *key-value pair*. JSON terdiri dari 2 struktur:

a. Kumpulan *key-value pair*. Di beberapa bahasa pemrograman lain, *key-value pair* direalisasikan sebagai objek, *struct*, *array* asosiatif, *record, dictionary*, dan lain sebagainya.

b. Daftar nilai terurut. Dalam bahasa pemrogram lain, biasanya berupa *array, vector, list,* atau *sequence*.

Kedua struktur tersebut bersifat universal, hampir semua bahasa pemrograman modern mendukungnya. Hanya saja nama sebutannya bermacam-macam. Oleh karena itu, JSON sudah tidak asing lagi di dunia pemrograman karena dibuat berdasarkan kedua struktur tersebut (JSON Team, 2015).

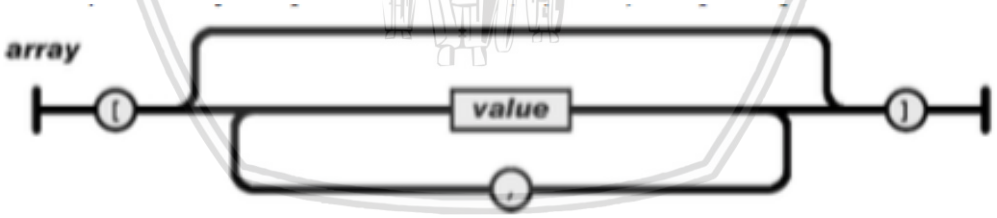
2.6.1 Struktur Objek JSON



Gambar 2.2 Struktur objek JSON (JSON Team, 2015)

Objek adalah suatu set dari *key-value pair*. Objek biasanya dimulai dengan tanda kurung kurawal kiri ({) dan diakhiri dengan tanda kurung kurawal kanan (}). Di dalamnya terdapat pasangan *key-value* yang dipisahkan dengan tanda titik dua (:). Jika ada lebih dari satu pasangan *key-value*, dipisahkan dengan tanda koma (,) seperti pada Gambar 2.2 (JSON Team, 2015).

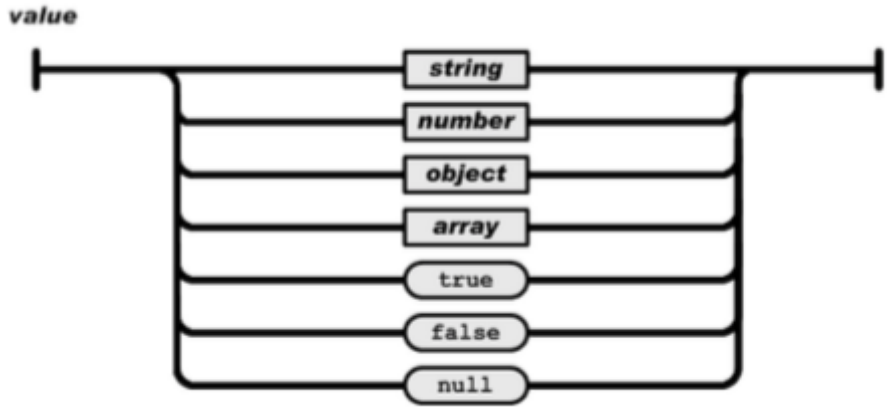
2.6.2 Struktur *Array* JSON



Gambar 2.3 Struktur *array* JSON (JSON Team, 2015)

Array adalah daftar nilai yang terurut. *Array* dimulai dengan tanda kurung siku kiri ([ ) dan diakhiri dengan tanda kurung siku kanan ( ]). Sama seperti objek, setiap pasangan *key-value* dipisahkan dengan tanda koma (,). *Array* dapat berisi *value* saja, jika *value* pada *array* dideklarasikan tanpa *key*, maka secara otomatis memiliki *key* berupa angka *integer* terurut dimulai dari indeks 0 (JSON Team, 2015).

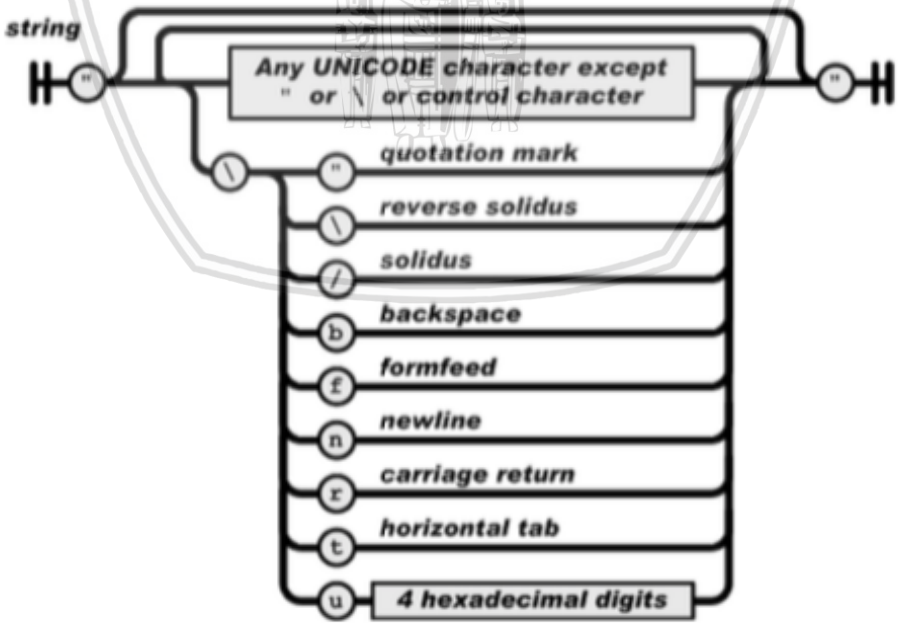
2.6.3 Tipe Data JSON



Gambar 2.4 Tipe data JSON (JSON Team, 2015)

Data dalam JSON dapat berupa berbagai macam tipe, antara lain *string*, boolean, angka, objek ataupun array seperti pada Gambar 2.4. Data *string* merupakan data teks, penulisannya dalam JSON diapit dengan tanda petik ("). Data boolean merupakan data dengan nilai benar (*true)*, salah (*false)*, atau kosong (*null)*. Data angka mencakup bilangan riil bertanda (termasuk bilangan negatif). JSON memperbolehkan data objek atau *array* bertingkat (objek/*array* dalam objek/*array*) (JSON Team, 2015).

2.6.3.1 Tipe Data *String*

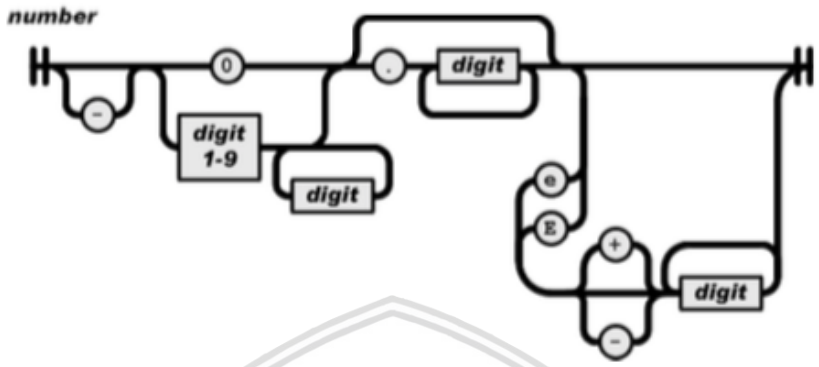


Gambar 2.5 Struktur data *string* dalam JSON (JSON Team, 2015)

String merupakan rangkaian karakter Unicode berjumlah nol atau lebih, yang diapit dengan tanda baca petik ganda ("), dengan *backslash escapes* seperti pada

Gambar 2.5. Satu karakter dalam JSON direpresentasikan sebagai satu karakter *string*. Format penulisannya sangat mirip dengan data *string* dalam bahasa pemrograman C atau Java (JSON Team, 2015).

2.6.3.2 Tipe Data Numerik

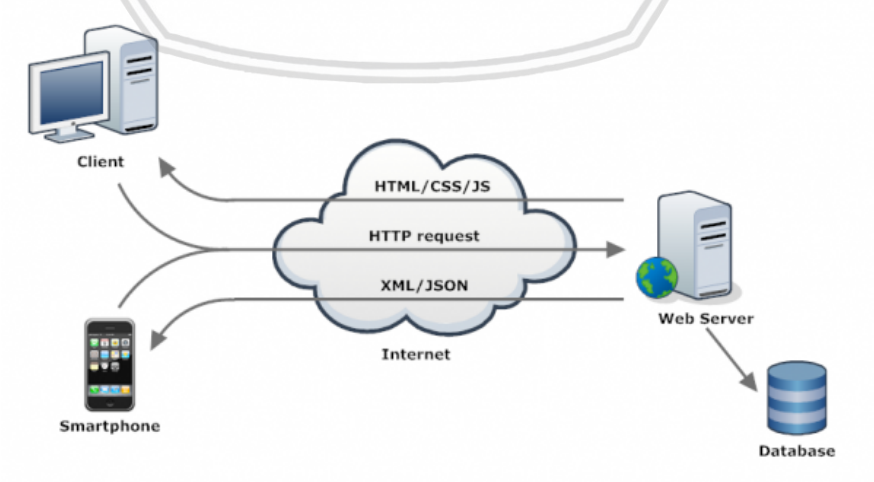


Gambar 2.6 Struktur data numerik dalam JSON (JSON Team, 2015)

Seperti halnya *string*, penulisan data numerik dalam JSON juga sangat mirip dengan data numerik dalam bahasa pemrograman C maupun Java. Hanya saja dalam JSON basis angka yang digunakan hanya basis desimal seperti yang ditampilkan pada Gambar 2.6. JSON mendukung penulisan notasi eksponensial (JSON Team, 2015).

2.6.4 *Web Service*

Web service merupakan komponen perangkat lunak yang berfungsi untuk komunikasi pertukaran data menggunakan standar teknologi Web, termasuk HTTP dan pesan berbasis XML (JSON Team, 2015). *Web service* dirancang agar dapat diakses untuk berbagai macam penggunaan mulai dari akses yang kompleks sampai yang sederhana, seperti mengecek saldo akun bank atau manajemen data dalam perancangan sumber daya perusahaan (Atzeni, 2011).



Gambar 2.7 Skema pertukaran data *web service* (Atzeni, 2011)

Pada Gambar 2.7 menunjukkan secara garis besar proses pertukaran data pada *web service*. *Web service* dijalankan pada komputer *server* yang bertugas untuk mengirim respon atas permintaan *client*. *Client* melakukan permintaan data dalam *web service* tidak harus melalui web yang diakses pada komputer, tetapi bisa juga melalui aplikasi pada suatu platform *smartphone* (Atzeni, 2011).

Ada dua kelas utama *web service*, yaitu (W3C, 2015):

1. REST-*compliant Web services*, di mana tujuan utamanya untuk memanipulasi representasi dari sumber daya Web dengan operasi yang bersifat *stateless*.
2. *Arbitrary Web services*, di mana proses pertukaran data dapat melibatkan beberapa set operasi tertentu.

Berikut beberapa keuntungan yang didapat dari penggunaan *web service*:

1. Format penggunaan terbuka untuk berbagai macam platform
2. Relatif mudah dimengerti sehingga mudah untuk melakukan debug
3. Dukungan *interface* yang stabil
4. Menggunakan pendekatan modular
5. Mudah untuk mengembangkan dengan transpor semantik tambahan
6. Biaya implementasi dan integrasi relatif murah

2.7 *REST Service*

REST singkatan dari *Representational State Transfer* pertama kali didiskusikan di acara disertasi Roy Fielding pada tahun 2000. Semenjak itu, *REST* mulai popular dan diterapkan pada beberapa area. Poin utama dari *REST* adalah kesederhanaan dan umumnya bergantung pada protokol HTTP dan perintah-perintah yang ada di dalamnya (NSA, 2011). Kunci dari penggunaan *REST* yang baik dan benar adalah kemampuan untuk memahami *data set*, mengidentifikasi kunci dari sumber daya, hubungan di antara kunci sumber daya, operasi yang diperbolehkan pada setiap sumber daya, dan kemampuan pengalamatan data menggunakan URI.

REST bukan merupakan standar atau protokol, melainkan prinsip/paradigma arsitektur perangkat lunak. Prinsip dasar dari *REST* adalah untuk menyederhanakan operasi, penamaan setiap sumber daya menggunakan URI, dan pemanfaatan perintah HTTP seperti GET, PUT, POST, DELETE berdasarkan cara penggunaan pada HTTP RFC (RFC 26163). *REST* bersifat *stateless*, tidak menyebutkan detail implementasi, dan interkoneksi antar sumber daya dilakukan melalui URI. *REST* juga dapat menggunakan perintah HTTP HEAD untuk untuk memeriksa sumber daya lain atau memperoleh *metadata* (NSA, 2011).

Berikut keuntungan dari penggunaan *REST*:

1. Setiap interkoneksi antar sumber daya diidentifikasi secar unik dan dapat dialamatkan menggunakan URI. (keuntungan konsistensi)

2. Hanya ada empat perintah HTTP yang digunakan, GET, PUT, POST, DELETE. (pemenuhan standar)
3. Data tidak dikirim secara langsung, tetapi tautan menuju ke data yang digunakan untuk melakukan akses, yang berguna untuk memperkecil beban jaringan, juga tetap memperbolehkan *repository* data untuk mengatur *access control*. (keuntungan kapasitas/efisiensi)
4. Dapat diimplementasikan secara cepat.
5. Dapat dipelajari dengan mudah, karena cara kerjanya mirip seperti World Wide Web sekarang ini.
6. Proxy dan firewall dapat disisipkan diantara *client* dan sumber daya.
7. *Stateless* menyederhanakan implementasi, tidak perlu mengatur urutan waktu.
8. Memfasilitasi integrasi dari layanan *RESTful* yang lain.
9. Memanfaatkan *client* untuk melakukan lebih banyak pekerjaan, karena *client* dapat juga dimanfaatkan sebagai sumber daya daripada harus membebankan semua proses ke *server*.

Beberapa kekurangan dari penggunaan *REST* antara lain:

1. *Server* atau *client* yang mengimplementasikan *REST* memiliki ancaman kelemahan yang sama dengan yang dimiliki oleh HTTP / aplikasi Web.
2. Jika perintah HTTP digunakan secara tidak benar, atau permasalahan dalam pemrosesan data tidak dapat diselesaikan menggunakan implementasi *RESTful*, kebanyakan menggunakan jalan keluar *Remote Procedure Call* (RPC) yang bukan merupakan solusi *RESTful*.
3. Sangat sedikit standar *library* yang mengakibatkan pihak pengembang melakukan lebih banyak pekerjaan dibanding menggunakan protokol berstandar yang memiliki *predefined library*.

Tabel 2.1 Kode data respon HTTP yang disarankan pada *REST* (NSA, 2011)

Sumber Daya	Kolektif contoh: api.contoh.com/resources	Satuan contoh: api.contoh.com/res/item17
GET	(200) *OK*, daftar data dari sumber daya yang dituju	(200) *OK*, data item satuan (404) *Not Found*, data tidak ditemukan jika ID salah
PUT	(404) *Not Found*, kecuali jika memang ingin mengganti seluruh data yang ada	(200) *OK*, jika data berhasil diubah (404) *Not Found*, jika ID tidak ditemukan (202) *No Content*, jika data kosong

Tabel 2.1 Kode data respon HTTP yang disarankan pada *REST* (lanjutan)

POST	(201) *Created*, jika daftar data berhasil dibuat	(404) *Not Found*, jika tidak ditemukan (409) *Conflict*, jika data yang ingin ditambahkan sudah ada
DELETE	(404) *Not Found*, kecuali jika memang ingin menghapus seluruh data yang ada	(200) *OK*, jika data berhasil dihapus (404) *Not Found*, jika ID tidak ditemukan

2.8 Firebase Realtime Database

Firebase Realtime Database (RDB) merupakan suatu layanan yang disediakan oleh Google. Firebase RDB menyediakan layanan untuk menyimpan data dalam format NOSQL *cloud database* yang dapat tersinkron secara *realtime* pada semua perangkat yang menggunakannya. Fitur-fitur yang disediakan oleh layanan Firebase RDB dapat dilihat pada Tabel 2.2 (Google Developers, 2017).

Tabel 2.2 Fitur layanan Firebase RDB (Google Developers, 2017)

Realtime	Firebase RDB menggunakan sinkronisasi data setiap kali ada perubahan data pada setiap perangkat yang tersambung. Sehingga dapat memudahkan pengembang perangkat lunak untuk membuat suatu sistem tanpa memikirkan tentang kode jaringan.
Offline	Aplikasi perangkat lunak yang mengimplementasikan Firebase tetap dapat dioperasikan dalam kondisi *offline*. SDK Firebase RDB menyimpan data secara lokal pada perangkat pengguna. Ketika perangkat tersambung kembali, Firebase RDB mensikronkan data secara otomatis dengan data yang berada pada server.
Dapat diakses pada berbagai perangkat *client*	Layanan Firebase RDB dapat secara langsung diakses melalui perangkat *client*, baik *smartphone* maupun web. Sehingga sistem aplikasi tidak memerlukan adanya *server*. Untuk aturan keamanan baca-tulis data dapat diatur pada *console platform* Firebase RDB.

Firebase RDB bekerja dengan cara menyediakan sambungan akses yang aman menuju *database* secara langsung dari perangkat *client*. Selain disimpan pada *server*, data juga disimpan pada perangkat secara lokal, proses sinkronisasi data akan dilakukan ketika perangkat pengguna memiliki akses jaringan internet.

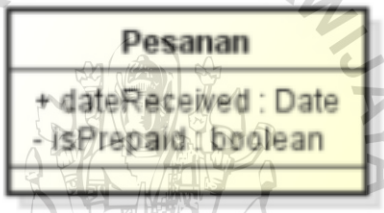
Firebase RDB dapat menggabungkan dan menyelesaikan konflik transaksi data secara otomatis ketika perangkat melakukan transaksi data pada waktu *offline* (Google Developers, 2017).

2.9 *Unified Modeling Language* (*UML*)

Unified Modeling Language (*UML*) merupakan notasi grafis dengan dukungan meta-model tunggal untuk membantu pendeskripsian desain dari suatu sistem perangkat lunak, khususnya pada sistem yang dibangun menggunakan pemrograman berorientasi objek (*object-oriented*). Pengertian *UML* dapat berbeda menurut pandangan masing-masing (Martin, 2010).

2.9.1 *Class Diagram*

Class merupakan cetakan dari objek dalam sistem dan memiliki berbagai macam hubungan statis yang saling berkaitan. Diagram *class* juga dapat berisi properti, operasi sebuah *class*, dan batasan-batasan yang terdapat dalam hubungan antar objek. Diagram *class* menggunakan istilah fitur untuk menggambarkan properti dan operasi sebuah *class* (Martin, 2010).



Gambar 2.8 Contoh diagram *class* (Martin, F. 2010)

Pada gambar 2.8 menunjukkan properti dari suatu *class* bernama "Pesanan", *dateReceived* adalah fungsi yang dimiliki oleh *class* tersebut. Sementara *Date* merupakan tipe data kembalian dari fungsi *dateReceived*. Simbol positif (+) disampingnya menandakan jika fungsi tersebut dapat diakses oleh kelas lain atau biasa disebut *public class*. Sedangkan simbol negatif (-) menandakan jika fungsi dalam *class* tersnut hanya dapat diakses oleh *class* itu sendiri (Martin, 2010).

2.9.2 Use Case

Use case merupakan teknik penggambaran fitur dan persyaratan fungsional dari suatu sistem. Diagram *use case* terdiri dari skenario-skenario yang merupakan langkah-langkah untuk menjabarkan sebuah interaksi antara pengguna dengan sistem. Pengguna biasa disebut sebagai aktor, dapat menggunakan lebih dari satu *use case* (Martin, 2010).

2.9.3 Activity Diagram

Activity diagram adalah diagram yang menggambarkan logika prosedural, proses bisnis, dan jalan kerja suatu sistem. *Activity diagram* memiliki peran yang menyerupai *flowchart* (diagram alir). Perbedaan utama dari *activity diagram*

dengan *flowchart* adalah prinsip notasi yang digunakan, di mana *activity diagram* mendukung penggunaan *behaviour* secara paralel (Martin, 2010).

2.9.4 *Sequence Diagram*

Dalam pemrograman berorientasi objek, interaksi antar objek dapat digambarkan menggunakan *sequence diagram*. Interaksi dapat berupa skenario dan alur dari proses-proses yang sedang berjalan. Pada umumnya alur *sequence diagram* diawali dengan *input* dan menghasilkan *output*. Selain itu juga digambarkan secara runtut proses apa saja yang terjadi diantara *input* dan *output*. *Sequence diagram* berbentuk grafik yang memiliki kolom-kolom horizontal. Pada tiap kolom, terdapat garis vertikal. Kolom-kolom *sequence diagram* mewakili *classifier* suatu objek, dengan garis vertikal mulai dari atas ke bawah mewakili waktu eksekusi suatu proses (Martin, 2010).

2.10 Pengujian Perangkat Lunak

Perangkat lunak yang dikembangkan berdasarkan suatu rancangan, memerlukan pengujian untuk dapat menemukan segala macam kemungkinan terjadinya kesalahan dan ketidaksesuaian dengan rancangan yang telah ditentukan sebelumnya (Simarmata, 2010). Pengujian perangkat lunak meliputi beberapa tingkat, salah satunya pengujian tingkat tinggi. Pengujian tingkat tinggi dilakukan dengan cara melakukan validasi secara rinci berdasarkan fungsi-fungsi utama suatu sistem yang telah dirancang sebelumnya (Pressman, 2001). Aplikasi *mobile geotagging* laporan kerusakan jalan memerlukan pengujian validasi dan pengujian *usability* agar tujuan dari penilitian dapat tercapai.

2.10.1 Pengujian Validasi

Untuk dapat memastikan semua persyaratan fungsional yang telah dirancang sebelumnya pada sistem telah diimplementasikan, dapat dilakukan pengujian validasi. Pengujian validasi meliputi skenario uji fungsional dan pengecekan persyaratan kinerja (Pressman, 2001). Dalam penelitian (Simarmata, 2010) (Simarmata, 2010) ini, pengujian validasi dilakukan menggunakan teknik pengujian *blackbox*. Pengujian *blackbox* dilakukan dengan cara membandingkan kesesuaian sistem aplikasi yang telah dibangun dengan rancangan kebutuhan.

2.10.2 Pengujian *Usability*

Pengujian *usability* dilakukan untuk mengetahui tingkat kemudahan dari penggunaan suatu sistem perangkat lunak. Penggunaan yang dimaksud meliputi efektifitas, efisiensi, dan tingkat kepuasan dalam suatu konteks tertentu (Rahardi, 2014). Cakupan pengujian *usability* dapat diukur dari lima komponen berikut (Nielsen, 2012):

1. *Learnability*

   Dilihat dari seberapa cepat pengguna dapat mengenali sistem dan dapat memulai untuk memfungsikannya (Nielsen, 2012).

2. *Efficiency*

   Dilihat dari tingkat kemudahan dan efektifitas untuk melakukan tugas pada suatu sistem aplikasi (Nielsen, 2012).

3. *Memorability*

   Seberapa mudah untuk mengingat proses pengerjaan suatu tugas yang telah dilakukan sebelumnya (Nielsen, 2012).

4. *Errors*

   Seberapa banyak kesalahan yang dilakukan oleh pengguna ketika menjalankan suatu fungsi/tugas (Nielsen, 2012).

5. *Satisfaction*

   Tingkat kepuasan dari penggunaan sistem secara keseluruhan yang berupa ukuran subjektif sebagaimana yang dirasakan oleh pengguna (Nielsen, 2012).

Dalam penelitian ini, acuan kuesioner yang dipakai adalah kuesioner USE. Kuesioner USE merupakan media kuesioner untuk mengukur *usability* dengan memakai 3 parameter yaitu kegunaan (*usefullness*), kepuasan (*satisfaction*) dan kemudahan penggunaan (*ease of use*). *Ease of use* merupakan sebuah parameter yang dibagi menjadi 2 faktor yaitu kemudahan dalam penggunaan (*ease of use*) dan kemudahan dalam mempelajari aplikasi (*ease of learning*) (Aelani & Falahah, 2012).

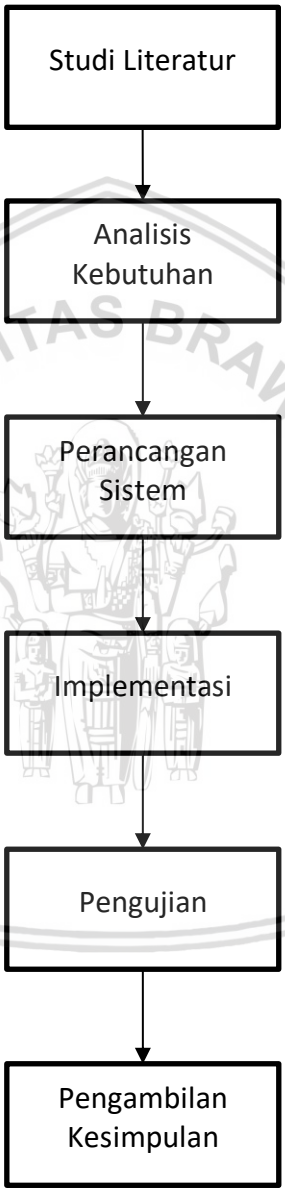
Contoh beberapa pertanyaan dalam kuesioner USE adalah sebagai berikut:

1. *Usefulness*
   - Aplikasi ini dalam pengerjaanya memenuhi ekspektasi saya.
   - Aplikasi ini membuat saya menjadi lebih produktif.
   - Aplikasi ini sangat berguna.
2. *Ease of Use*
   - Aplikasi ini mudah digunakan.
   - Aplikasi ini *user-friendly*.
   - Aplikasi ini fleksibel.
3. *Ease of Learning*
   - Aplikasi ini dapat dengan mudah dan cepat saya pelajari.
   - Aplikasi ini mudak diingat dalam penggunaanya.
4. *Satisfaction*
   - Aplikasi ini menyenangkan untuk digunakan.
   - Saya merasa saya harus memiliki aplikasi ini.

Skala Likert dalam penelitian ini digunakan sebagai acuan penilaian dan analisis hasil dalam melakukan survei untuk keperluan pengujian *usability*. Metode Likert merupakan suatu metode penentuan skala dalam pernyataan sikap yang menggunakan distribusi respons sebagai dasar penentuan nilai skalanya. Nilai dari skala Likert tergantung dari suatu kebutuhan. Skala Likert menjabarkan variabel yang diukur menjadi indikator variabel dimana indikator tersebut kemudian

BAB 3 METODOLOGI PENELITIAN

Pada bab ini dijelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi, dan pengujian dari aplikasi perangkat bergerak yang akan dikembangkan. Kesimpulan dan saran diikutsertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya. Gambar 3.1 menampilkan diagram alir dari tahapan-tahapan pengerjaan penelitian ini.



Gambar 3.1 Diagram alir metodologi penelitian

3.1 Studi Literatur

Dalam penelitian ini, diperlukan literatur acuan sebagai dasar teori yang digunakan untuk menunjang penulisan. Berikut merupakan daftar acuan pustaka dan teori pendukung yang digunakan:

1. Aplikasi Perangkat Bergerak
2. Bahasa Pemrograman Java
3. *Javascript Object Notation*
4. Android SDK
5. Geotagging
6. Google Maps API
7. Firebase *Realtime Database*

3.2 Analisis Kebutuhan

Tujuan analisis kebutuhan untuk merangkum semua kebutuhan yang diperlukan sistem perangkat lunak yang akan dikembangkan. Kegiatannya meliputi analisis spesifikasi perangkat lunak. Metode analisis yang digunakan adalah UML (*Unified Modeling Language*) yang merupakan analisis berorientasi objek. *Use case diagram* digunakan untuk mendeskripsikan perincian kebutuhan dan fungsionalitas sistem dari sudut pandang pengguna. Kemudian analisis kebutuhan dilakukan untuk mengidentifikasi kebutuhan sistem aplikasi perangkat bergerak *geotagging* laporan kerusakan jalan. Kebutuhan fungsionalitas dari aplikasi antara lain:

1. Aplikasi perangkat bergerak dapat mengirimkan laporan kerusakan jalan dengan foto dan data lokasi menggunakan *geotagging*.
2. Sistem dapat menampilkan hasil laporan pengguna.
3. Aplikasi dapat menampilkan status laporan kerusakan jalan.

3.3 Perancangan Sistem

Perancangan aplikasi merupakan tahapan setelah analisis kebutuhan. Pada tahap ini, dilakukan identifikasi fitur aplikasi yang dimodelkan dalam *use case diagram.* Dari hasil *use case diagram* dilakukan perancangan kelas yang dimodelkan dalam bentuk *class diagram*. Selanjutnya interaksi antar objek diidentifikasi dan dimodelkan menggunakan *sequence diagram* yang menggambarkan urutan waktu dari interaksi antar objek. Ada pula perancangan basis data yang dimodelkan menggunakan skema data JSON dan juga perancangan *model interface* menggunakan *page flow*.

3.4 Implementasi

Dari hasil perancangan yang telah dibuat akan diimplementasikan pada pengembangan sistem. Implementasi sistem akan diterapkan menggunakan bahasa Java pada *platform* Android dan HTML, CSS, Javascript pada *platform web* untuk admin. Pada *platform* Android, pengembangan aplikasi menggunakan

Integrated Development Environment (IDE) Android Studio, sedangkan pada *platform web* untuk admin menggunakan Visual Studio Code.

3.5 Pengujian dan Analisis

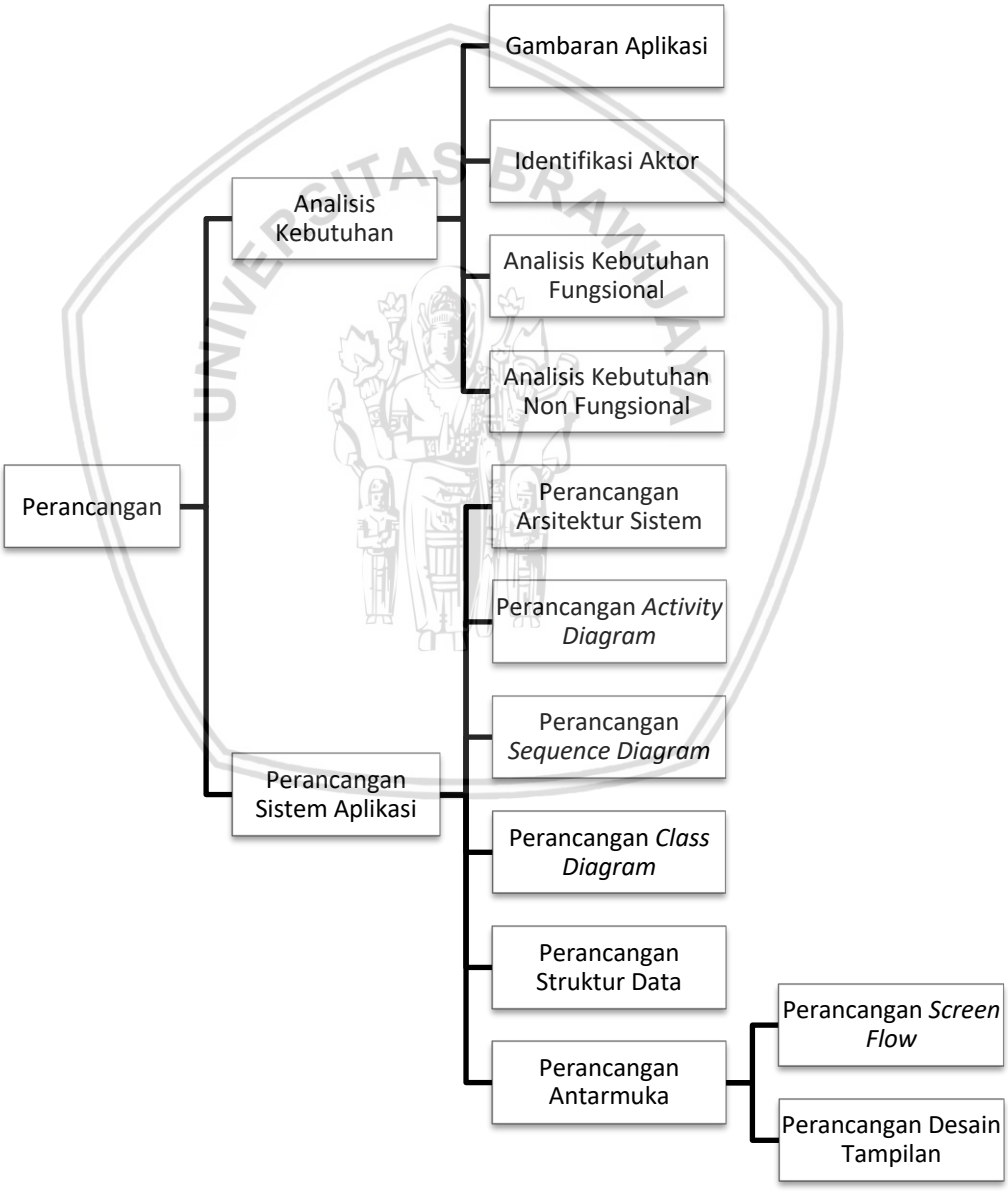
Pengujian dilakukan untuk menunjukkan bahwa sistem dapat bekerja sesuai dengan spesifikasi dan dapat memenuhi kebutuhan penggunanya. Selain itu, juga untuk mengetahui kinerja dan performa aplikasi. Pada umumnya pengujian menggunakan teknik *blackbox* untuk melakukan validasi dari fungsionalitas sistem. Selain fungsionalitas, juga dilakukan pengujian usabilitas menggunakan metode kuesioner. Kuesioner yang dibuat berdasarkan metode skala *likert* di mana responden memberikan nilai atau skor 1 sampai 5. Pertanyaan yang diajukan dibuat berdasarkan pihak sebelumnya yang telah melakukan kuesioner terhadap aplikasi perangkat bergerak dengan pengujian usabilitas yang sejenis.

3.6 Kesimpulan dan Saran

Tahap kesimpulan dan saran dilakukan setelah tahap perancangan perangkat lunak, implementasi perangkat lunak, dan pengujian perangkat lunak telah selesai dikerjakan. Kesimpulan diperoleh dari hasil pengujian dan analisis terhadap sistem perangkat lunak yang dibangun yang dirumuskan dari rumusan masalah. Sedangkan saran dimaksudkan untuk memperbaiki kesalahan-kesalahan dan menyempurnakan penelitian serta untuk menjadi pertimbangan dalam pengembangan aplikasi selanjutnya.

BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN

Pada bab ini menjelaskan tentang kebutuhan dalam merancang sistem aplikasi *mobile geotagging* kerusakan jalan dengan memperhatikan kemudahan penggunaan aplikasi bagi pengguna. Analisis kebutuhan mencakup proses bisnis aplikasi, identifikasi aktor, kebutuhan fungsional yang kemudian akan digambarkan dengan diagram *use case*. Dilanjutkan dengan tahap perancangan. Tahap perancangan dibagi menjadi beberapa bagian yaitu, perancangan arsitektur, diagram aktifitas (*activity diagram*), dan pemodelan diagram kelas (*class diagram*). Langkah-langkah rancangan aplikasi *mobile geotagging* laporan kerusakan jalan direpresentasikan dalam Gambar 4.1.



Gambar 4.1 *Tree diagram* perancangan sistem

4.1 Analisis Kebutuhan

Tahap analisis kebutuhan dilakukan dengan cara melakukan identifikasi seluruh kebutuhan aplikasi *mobile geotagging* kerusakan jalan. Analisis kebutuhan bertujuan untuk menggambarkan kebutuhan-kebutuhan yang harus disediakan sistem agar dapat memenuhi kebutuhan pengguna. Tahap-tahap analisis kebutuhan meliputi gambaran umum aplikasi, proses bisnis aplikasi, identifikasi aktor, kebutuhan fungsional sistem yang kemudian diimplementasikan ke dalam diagram *use case*, dan kebutuhan non- fungsional sistem.

4.1.1 Gambaran Umum Sistem

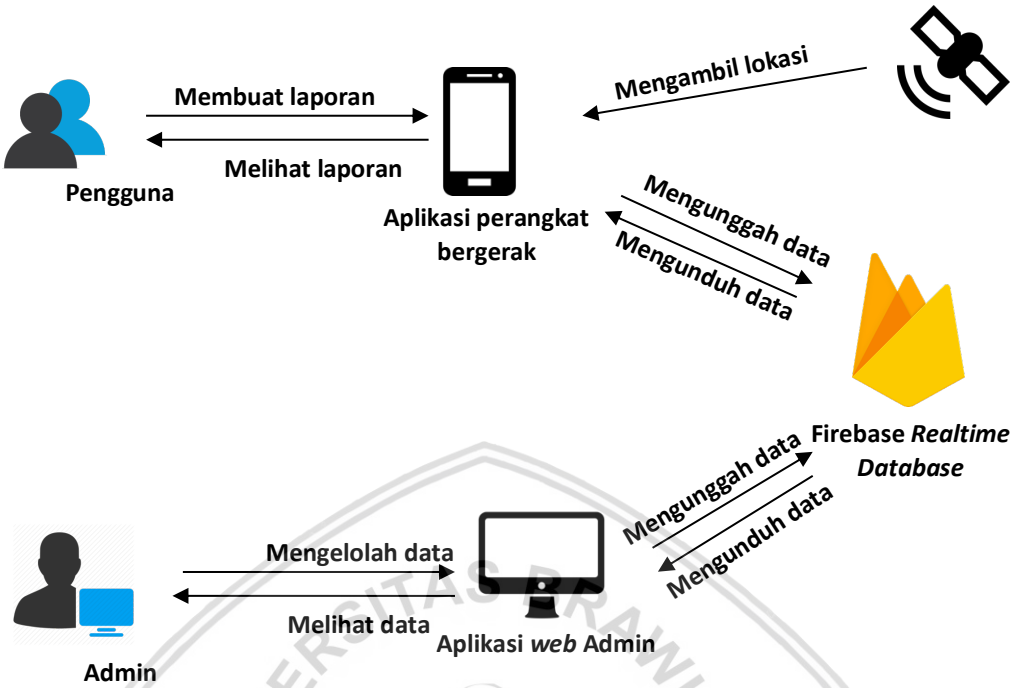
Aplikasi *mobile geotagging* kerusakan jalan bertujuan untuk mempermudah masyarakat dalam melaporkan keluhan kerusakan jalan. Selain untuk melaporkan, melalui aplikasi ini masyarakat juga dapat melihat data laporan kerusakan jalan yang telah dilaporkan.

Sistem aplikasi ini nantinya akan terbagi menjadi 2 bagian. Aplikasi *mobile client* untuk digunakan oleh masyarakat sebagai pihak pelapor kerusakan jalan, dan *web front-end* digunakan oleh admin sistem pelaporan kerusakan jalan untuk mengelolah data laporan. Adapun *back-end server* memanfaatkan teknologi Firebase yang menyediakan layanan *realtime database*.

Pada bagian *mobile client*, aplikasi ini memiliki fitur untuk membuat laporan laporan kerusakan jalan yang berisikan data foto, data lokasi, dan keterangan yang bersifat opsional. Setelah data laporan tersebut lengkap, pengguna dapat mengunggah data laporan. Pengguna juga dapat melihat visualisasi data laporan pada tampilan peta dan berinteraksi dengan data laporan dari pengguna lain dengan cara menekan tanda bintang pada laporan tersebut untuk menaikkan bobot laporan. Nilai tersebut nantinya akan digunakan untuk menentukan prioritas kerusakan jalan dan bobot visualisasi data laporan yang ditampilkan dalam bentuk *heat map*. Selain itu, pengguna juga dapat mengisukan jalan yang telah diperbaiki dan mengisukan penyalahgunaan laporan pengguna lain, yang nantinya akan ditangani oleh admin. Untuk mengakses semua fitur tersbut, pengguna harus melakukan login terlebih dahulu menggunakan nomor telepon tanpa memerlukan *password*. Pada nomor telepon yang dimasukkan, akan dikirimkan pesan singkat berisi nomor verifikasi untuk dimasukkan kedalam form verifikasi. Jika nomor yang dimasukkan adalah nomor pada perangkat *smartphone* yang sama, kode verifikasi dapat dimasukkan secara otomatis oleh aplikasi.

Pada bagian *web front-end*, admin harus melakukan login untuk dapat mengakses antarmuka pengelolahan laporan dan pengguna. Terdapat fitur visualisasi data laporan, melihat laporan penyalahgunaan, mengubah status laporan rusak menjadi diperbaiki atau sebaliknya, menghapus laporan yang disalahgunakan, juga fitur untuk memblokir pengguna yang melakukan penyalahgunaan.

4.1.2 Proses Bisnis Aplikasi



Gambar 4.2 Proses bisnis sistem pelaporan

Gambar 4.2 menunjukkan proses bisnis dari sistem aplikasi *geotagging* laporan kerusakan jalan secara umum. Terdapat dua aktor dalam proses bisnis sistem aplikasi, yaitu masyarakat sebagai pengguna aplikasi *mobile* dan admin sebagai pengguna aplikasi *web*. Sistem aplikasi terdiri dari dua bagian yaitu aplikasi *mobile* pada perangkat Android, halaman admin pada web. Sedangkan *back end server* menggunakan teknologi Firebase yang menyediakan layanan *backend as a service (BaaS)* yaitu *realtime-database* dan Firebase *storage.*

Pengguna dapat membuat laporan baru untuk kemudian diunggah ke server dan dapat melihat laporan yang telah diunggah. Data yang ditampilkan pada aplikasi berupa data *timeline* daftar laporan, laporan oleh pengguna itu sendiri, visualisasi data berupa marker, dan visualisasi data berupa *heat map* yang menggunakan rating sebagai nilai pembobotan intensitas. Untuk interaksi pengguna dengan data laporan, pengguna dapat memberikan *up vote* dengan menekan tombol bintang untuk meningkatkan bobot laporan. Pengguna juga dapat mengisukan jika jalan yang dilaporkan rusak telah diperbaiki maupun isu tentang penyalahgunaan laporan dari pengguna lain. Dari isu yang diajukan oleh pengguna, dapat ditangani oleh admin dengan cara mengubah data status laporan kerusakan jalan, menghapus data laporan yang disalahgunakan, maupun memblokir akun pengguna berdasarkan tingkat penyalahgunaan.

Pada halaman admin, terdapat visualisasi data laporan seperti pada aplikasi *mobile*, tabel pengajuan isu jalan telah diperbaiki, laporan penyalahgunaan, dan fitur download data laporan kerusakan jalan dalam format tertentu. Admin tidak

memiliki akses untuk mengubah data laporan maupun membuat laporan baru, admin hanya dapat menghapus data laporan kerusakan jalan berdasarkan laporan penyalahgunaan oleh pengguna. Jika dinilai tingkat penyalahgunaan cukup parah, admin dapat memblokir ID pengguna sehingga pengguna tidak dapat membuat laporan baru.

4.1.3 Identifikasi Aktor

Untuk mengetahui siapa saja aktor yang terlibat secara langsung dalam sistem aplikasi *geotagging* laporan kerusakan jalan yang akan dibuat. Pada sistem ini terdapat dua aktor yaitu pengguna dan admin. Untuk detail penjelasan masin-masing aktor ditunjukkan pada Tabel 4.1

Tabel 4.1 Identifikasi aktor

Kategori Aktor	Deskripsi
Pengguna	Merupakan aktor pada aplikasi *mobile* Android, dapat menggunakan aplikasi dengan cara melakukan proses *login* terlebih dahulu menggunakan nomor telepon pengguna. Sistem akan melakukan pendaftaran dan login otomatis untuk mempermudah dan mempercepat pengoperasian aplikasi. Pengguna dapat memperoleh informasi data laporan, juga dapat membuat laporan baru dan mengunggahnya langsung jika kondisi memungkinkan, atau menyimpan foto, dan mengunggah laporan dengan menggunakan foto yang tersimpan pada galeri.
Admin	Merupakan aktor pada aplikasi web, memiliki hak akses untuk melihat visualisasi data laporan kerusakan jalan, data laporan penyalahgunaan, dapat menghapus data laporan. Admin juga dapat mengubah hak akses pengguna berdasarkan isu penyalahgunaan aplikasi pada sisi *client*.

4.1.4 Analisis Kebutuhan Fungsional Sistem

Kebutuhan fungsional merupakan spesifikasi daftar fitur sistem yang dibutuhkan oleh pengguna. Pada sistem sistem *geotagging* laporan kerusakan jalan ini tedapat 2 bagian kebutuhan fungsional, yaitu pada aplikasi perangkat bergerak laporan *geotagging* kerusakan jalan dan web *front-end* untuk pengelolahan data oleh admin. Pada Tabel 4.2 dan Tabel 4.3 menjelaskan tentang daftar dari masing-masing kebutuhan fungsional.

Tabel 4.2 Kebutuhan fungsional aplikasi perangkat bergerak

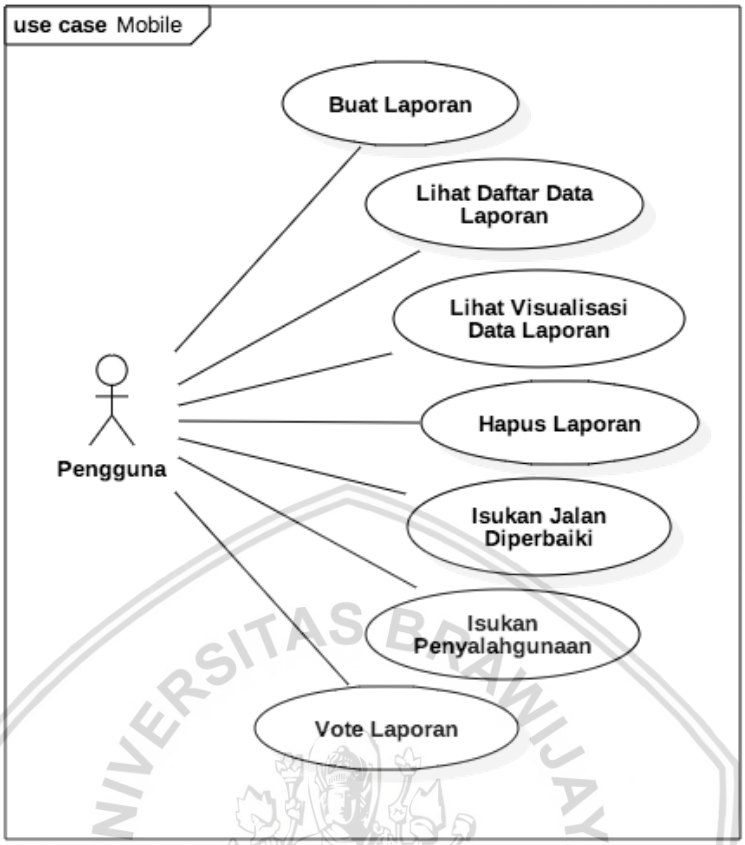
ID	Kebutuhan Fungsional	*Use Case*
SRS-M01	Sistem menyediakan fitur untuk mengunggah laporan	Buat Laporan
SRS-M02	Sistem menyediakan fitur untuk melihat daftar data laporan.	Lihat Daftar Data Laporan
SRS-M03	Sistem menyediakan fitur untuk melihat visualisasi data laporan pada tampilan peta berupa *clustered marker* dan *heat map*	Lihat Peta Visualisasi Data Laporan
SRS-M04	Sistem menyediakan fitur untuk menghapus laporan kerusakan jalan	Hapus Laporan
SRS-M05	Sistem menyediakan fitur untuk mengisukan laporan kerusakan jalan yang telah diperbaiki	Isukan Jalan Diperbaiki
SRS-M06	Sistem menyediakan fitur mengisukan penyalahgunaan laporan kerusakan jalan	Isukan Penyalahgunaan
SRS-M07	Sistem menyediakan fitur untuk *vote* laporan	*Vote* Laporan

Tabel 4.3 Kebutuhan fungsional aplikasi web

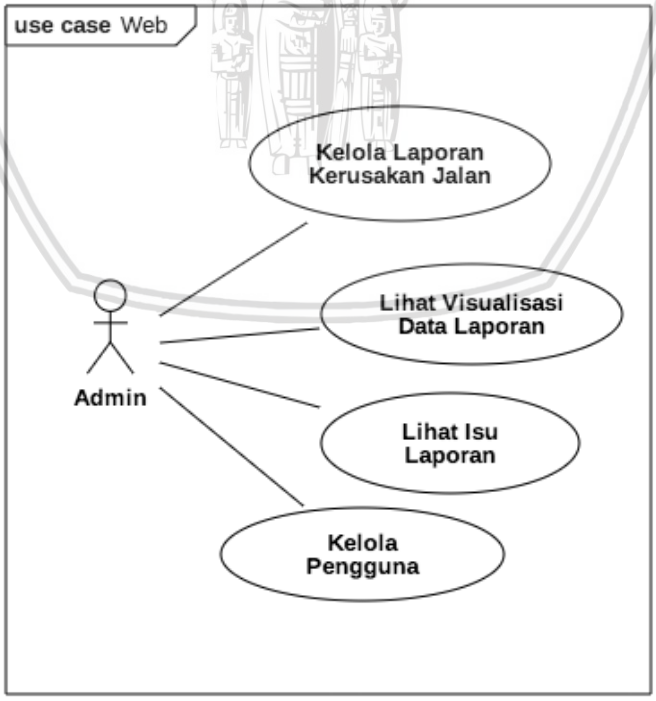
ID	Kebutuhan Fungsional	*Use Case*
SRS-W01	Sistem menyediakan fitur untuk mengelola laporan kerusakan jalan	Kelola Laporan Kerusakan Jalan
SRS-W02	Sistem menyediakan fitur untuk melihat visualisasi data laporan pada peta	Lihat Visualisasi Data Laporan
SRS-W03	Sistem menyediakan fitur untuk melihat isu jalan diperbaiki dan isu penyalahgunaan	Lihat Isu Laporan
SRS-W04	Sistem menyediakan fitur untuk melihat dan mengubah hak akses pengguna aplikasi *mobile*	Kelola Pengguna

4.1.4.1 Diagram *Use Case*

Diagram *use case* menggambarkan aktifitas apa saja yang dapat dilakukan oleh aktor pada suatu sistem. Gambar 4.3 dan Gambar 4.4 berikut merupakan diagram *use case* untuk sistem aplikasi *mobile geotagging* kerusakan jalan berdasarkan daftar analisis kebutuhan untuk sistem aplikasi perangkat bergerak dan sistem aplikasi web.



Gambar 4.3 Diagram *use case* sistem aplikasi perangkat bergerak



Gambar 4.4 Diagram *use case* sistem aplikasi web

4.1.4.2 Skenario *Use Case*

Skenario *use case* berikut menjelaskan lebih rinci untuk setiap fitur yang ada pada *use case*. Pada tabel skenario *use case* terdapat nama *use case*, *identifier use case*, aktor, tujuan, deskripsi, *pre-condition,* dan *post-condition*. Kemudian di bawahnya berisi tanggapan yang diberikan oleh sistem ketika melakukan aktifitas tersebut (*main flow*) dan tanggapan alternatif (*alternative flow)* jika terdapat percabangan kondisi aktifitas yang tidak terpenuhi.

Tabel 4.4 Skenario *Use Case* buat laporan

Nama *Use Case*	Buat Laporan
Use Case Identifier	SRS_M01
Aktor	Pengguna
Tujuan	Melihat daftar laporan kerusakan jalan
Deskripsi	Memungkin pengguna untuk membuat dan mengunggah laporan kerusakan jalan baru
Pre-Condition	Pengguna telah melakukan *login* dan berada pada tampilan utama
Post-Condition	Laporan kerusakan jalan baru ditampilkan pada daftar laporan
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol tambah di pojok kanan bawah pada tampilan utama	2. Menampilkan kamera/galeri gambar
3. Mengambil gambar dari kamera/memilih gambar dari galeri	4. Menampilkan pratinjau laporan kerusakan baru.
	5. Mengambil data koordinat lokasi dari EXIF foto
	6. Menerjemahkan data koordinat lokasi menjadi nama jalan dan menampilkannya
7. Menekan tombol unggah laporan di pojok kanan bawah pada tampilan pratinjau laporan baru	8. Mengunggah data laporan ke server dan memindahkan tampilan ke tampilan utama
***Alternative Flow*: data lokasi EXIF tidak ditemukan**	
	5a. Mengambil data koordinat lokasi dari sensor GPS

Tabel 4.4 Skenario *Use Case* buat laporan (lanjutan)

***Alternative Flow*: sensor GPS tidak diaktifkan/data koordinat GPS tidak ditemukan**	
	5b. Menampilkan pesan lokasi tidak ditemukan dan tombol atur lokasi manual
6b. Menekan tombol atur lokasi manual	7b. Menampilkan tampilan peta untuk memilih lokasi
8b. Memilih lokasi laporan jalan rusak pada tampilan peta dan menekan tombol konfirmasi di pojok kanan bawah	9b. Menerjemahkan data koordinat lokasi yang dipilih menjadi nama jalan
10b. Menekan tombol unggah laporan di pojok kanan bawah tampilan pratinjau laporan baru	11b. Mengunggah data laporan ke server dan memindahkan tampilan ke tampilan utama

Tabel 4.5 Skenario *Use Case* lihat daftar laporan

Nama *Use Case*	Lihat Daftar Laporan
Use Case Identifier	SRS_M02
Aktor	Pengguna
Tujuan	Melihat daftar laporan kerusakan jalan
Deskripsi	Memungkin pengguna untuk dapat melihat daftar data laporan kerusakan jalan yang diurutkan berdasarkan waktu.
Pre-Condition	Pengguna telah melakukan *login*
Post-Condition	Daftar *timeline* laporan kerusakan jalan ditampilkan
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Membuka Aplikasi	2. Mengunduh daftar laporan dari server
	3. Menampilkan daftar laporan kerusakan jalan yang telah diunduh dari server

Tabel 4.5 Skenario *Use Case* lihat daftar laporan (lanjutan)

***Alternative Flow*: akses server gagal**	
	3a. Menampilkan daftar laporan dari *cache*

Tabel 4.6 Skenario *Use Case* lihat visualisasi data laporan

Nama *Use Case*	Lihat Visualisasi Data Laporan
Use Case Identifier	SRS_M03
Aktor	Pengguna
Tujuan	Melihat visualisasi data laporan kerusakan jalan
Deskripsi	Memungkinkan pengguna untuk dapat melihat visualisasi data laporan kerusakan jalan pada tampilan peta dalam bentuk *marker* atau *heat map*.
Pre-Condition	Pengguna berada pada tampilan halaman utama (melihat daftar data laporan)
Post-Condition	Visualisasi data laporan kerusakan jalan ditampilkan
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol peta yang terletak pada *toolbar* halaman utama	2. Mengunduh data laporan dari server
	3. Menampilkan visualisasi data laporan yang telah diunduh dari server pada tampilan peta
***Alternative Flow*: akses server gagal**	
	3a. Menampilkan visualisasi data laporan dari *cache* pada tampilan peta

Tabel 4.7 Skenario *Use Case* hapus laporan

Nama *Use Case*	Hapus Laporan
Use Case Identifier	SRS_M04
Aktor	Pengguna
Tujuan	Menghapus laporan yang telah diunggah pengguna

Tabel 4.7 Skenario *Use Case* hapus laporan (lanjutan)

Deskripsi	Memungkinkan pengguna untuk dapat menghapus laporan yang telah diunggahnya, tetapi tidak untuk laporan yang diunggah oleh pengguna lain.
Pre-Condition	Pengguna berada pada tampilan halaman utama (melihat daftar data laporan)
Post-Condition	Laporan yang telah dihapus dihilangkan pada tampilan daftar data laporan kerusakan jalan

Main Flow	
Aksi Aktor	**Respon Sistem**
1. Pengguna menekan tombol menu pada laporan milik pengguna yang sedang *login*	2. Menampilkan menu laporan
3. Menekan menu *Delete Report*	4. Melakukan permintaan hapus laporan yang dipilih ke server dan menghilangkan laporan yang dihapus dari tampilan

Tabel 4.8 Skenario *Use Case* isukan jalan diperbaiki

Nama *Use Case*	Isukan Jalan Diperbaiki
Use Case Identifier	SRS_M05
Aktor	Pengguna
Tujuan	Mengisukan bahwa jalan yang dilaporkan rusak telah diperbaiki
Deskripsi	Memungkinkan pengguna mengisukan laporan jalan rusak yang telah diperbaiki untuk kemudian dapat diubah oleh admin status dari laporan tersebut.
Pre-Condition	Pengguna berada pada tampilan halaman utama (melihat daftar data laporan)
Post-Condition	Data isu jalan diperbaiki terkirim ke server dan tetap menampilkan halaman utama

Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol menu pada daftar laporan	2. Menampilkan menu laporan

Tabel 4.8 Skenario *Use Case* isukan jalan diperbaiki (lanjutan)

3. Memilih menu *Issue Fixed Road*	4. Mengirimkan data isu jalan telah diperbaiki ke server dan menutup menu laporan

Tabel 4.9 Skenario *Use Case* isukan penyalahgunaan

Nama *Use Case*	Isukan Penyalahgunaan
Use Case Identifier	SRS_M06
Aktor	Pengguna
Tujuan	Mengisukan laporan yang disalahgunakan oleh pengguna
Deskripsi	Memungkinkan pengguna mengisukan penyalahgunaan laporan pengguna lainnya untuk kemudian dapat dihapus oleh admin.
Pre-Condition	Pengguna berada pada tampilan halaman utama (melihat daftar data laporan)
Post-Condition	Data isu jalan diperbaiki terkirim ke server dan tetap menampilkan halaman utama
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol menu pada daftar laporan	2. Menampilkan menu laporan
3. Tekan menu *Issue Abusive Post*	4. Mengirimkan data isu jalan telah diperbaiki ke server dan menutup menu laporan

Tabel 4.10 Skenario *Use Case vote* laporan

Nama *Use Case*	*Vote* Laporan
Use Case Identifier	SRS_M07
Aktor	Pengguna
Tujuan	Memberikan *vote* pada laporan kerusakan jalan oleh pengguna lain

Tabel 4.10 Skenario *Use Case vote* laporan (lanjutan)

Deskripsi	Memungkinkan pengguna untuk dapat menaikkan nilai bobot laporan kerusakan jalan oleh pengguna lain.
Pre-Condition	Pengguna berada pada tampilan halaman utama (melihat daftar data laporan)
Post-Condition	Ikon tombol *vote* terisi warna kuning dan angka jumlah *vote* disamping ikon bertambah satu
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol ikon bintang	2. Mengirimkan data *vote* ke server
	3. Mengubah warna ikon tombol *vote* dan mengubah angka jumlah *vote*

Tabel 4.11 Skenario *Use Case* admin kelola laporan kerusakan jalan

Nama *Use Case*	Kelola Laporan Kerusakan Jalan
Use Case Identifier	SRS_W01
Aktor	Admin
Tujuan	Mengelola Laporan Kerusakan Jalan
Deskripsi	Memungkinkan admin untuk melihat daftar, mengubah status, dan menghapus laporan kerusakan jalan.
Pre-Condition	Aktor membuka halaman *web* admin pada browser dan telah melakukan proses *login*.
Post-Condition	Menampilkan halaman utama
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Mengakses aplikasi web admin	2. Menampilkan halaman utama yang berisi daftar laporan kerusakan jalan

Tabel 4.12 Skenario *Use Case* admin lihat visualisasi data laporan

Nama *Use Case*	Lihat Visualisasi Data Laporan
Use Case Identifier	SRS_W02

Tabel 4.12 Skenario *Use Case* admin lihat visualisasi data laporan (lanjutan)

Aktor	Admin
Tujuan	Melihat visualisasi data laporan kerusakan jalan
Deskripsi	Memungkinkan pengguna untuk dapat melihat visualisasi data laporan kerusakan jalan pada tampilan peta dalam bentuk *marker* atau *heat map*.
Pre-Condition	Aktor telah melakukan login
Post-Condition	Menampilkan peta dengan visualisasi data laporan kerusakan jalan berupa *clustered marker* atau *heat map*

Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol peta pada navigasi	2. Menampilkan tampilan peta dengan visualisasi data laporan kerusakan jalan

Tabel 4.13 Skenario *Use Case* admin lihat isu laporan

Nama *Use Case*	Lihat Isu Laporan
Use Case Identifier	SRS_W03
Aktor	Admin
Tujuan	Melihat isu laporan
Deskripsi	Memungkinkan admin untuk melihat laporan kerusakan jalan yang diisukan telah diperbaiki maupun isu penyalahgunaan.
Pre-Condition	Aktor telah melakukan login
Post-Condition	Menampilkan daftar isu jalan diperbaiki

Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol navigasi isu laporan	2. Menampilkan daftar isu laporan kerusakan jalan

Tabel 4.14 Skenario *Use Case* admin kelola pengguna

Nama *Use Case*	Kelola Pengguna
Use Case Identifier	SRS_W04
Aktor	Admin
Tujuan	Mengelola pengguna aplikasi *mobile*
Deskripsi	Memungkinkan admin untuk mengelola pengguna aplikasi *mobile* untuk melihat daftar dan mengubah hak akses pengguna
Pre-Condition	Aktor telah melakukan login
Post-Condition	Menampilkan daftar pengguna
Main Flow	
Aksi Aktor	**Respon Sistem**
1. Menekan tombol navigasi kelola pengguna	2. Menampilkan daftar pengguna beserta tombol untuk mengubah hak akses

4.1.5 Analisis Kebutuhan Non Fungsional

Untuk mengetahui bagaimana suatu sistem bekerja secara umum, dilakukan analisis kebutuhan non fungsional. Kebutuhan non fungsional pada aplikasi *mobile geotagging* pelaporan jalan rusak adalah *usability* dan *availability*. Tabel 4.15 berikut berisikan daftar kebutuhan non fungsional yang berisi penjalasannya.

Tabel 4.15 Kebutuhan non fungsional

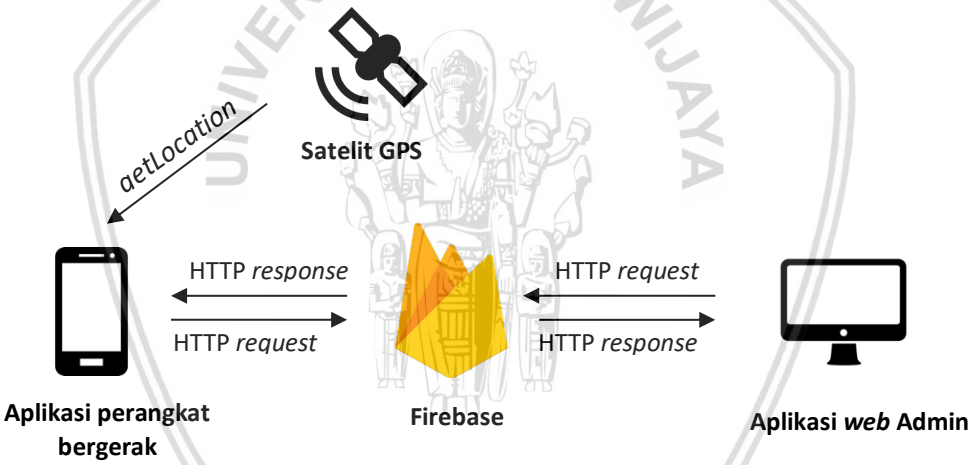
Parameter	Deskripsi Kebutuhan
Usability	Aplikasi dapat dioperasikan pengguna dengan mudah. Tampilan antarmuka dirancang sesederhana mungkin sehingga pengguna dapat memahami alur penggunaan aplikasi tanpa memerlukan panduan
Reliability	Aplikasi dapat diakses dan beroperasi dalam keadaan *offline* untuk melihat data dan visualisasi laporan

4.2 Perancangan Aplikasi Perangkat Bergerak

Berikut tahap-tahap perancangan aplikasi sosial *geotagging* kerusakan jalan yaitu perancangan arsitektur sistem, perancangan diagram, perancangan basis data, perancangan navigasi dan antarmuka aplikasi. Perancangan diagram terdiri dari perancangan *class diagram, activity diagram,* dan *sequence diagram*. Perancangan skema JSON pada Firebase *realtime database*.

4.2.1 Perancangan Arsitektur Sistem

Sistem aplikasi *mobile geotagging* laporan kerusakan jalan memiliki akses langsung ke *backend service* Firebase. Sistem aplikasi *mobile* juga dapat mengambil data koordinat lokasi pengguna dari Exif foto atau GPS untuk menyematkan data lokasi pada data foto yang akan diunggah (*geotaggingg*). Sama halnya dengan aplikasi *mobile*, aplikasi web admin dapat terhubung secara langsung ke server Firebase. Transaksi data antara *mobile client* / web admin dengan server Firebase menggunakan format JSON. Gambaran arsitektur sistem ditunjukan pada Gambar 4.5.



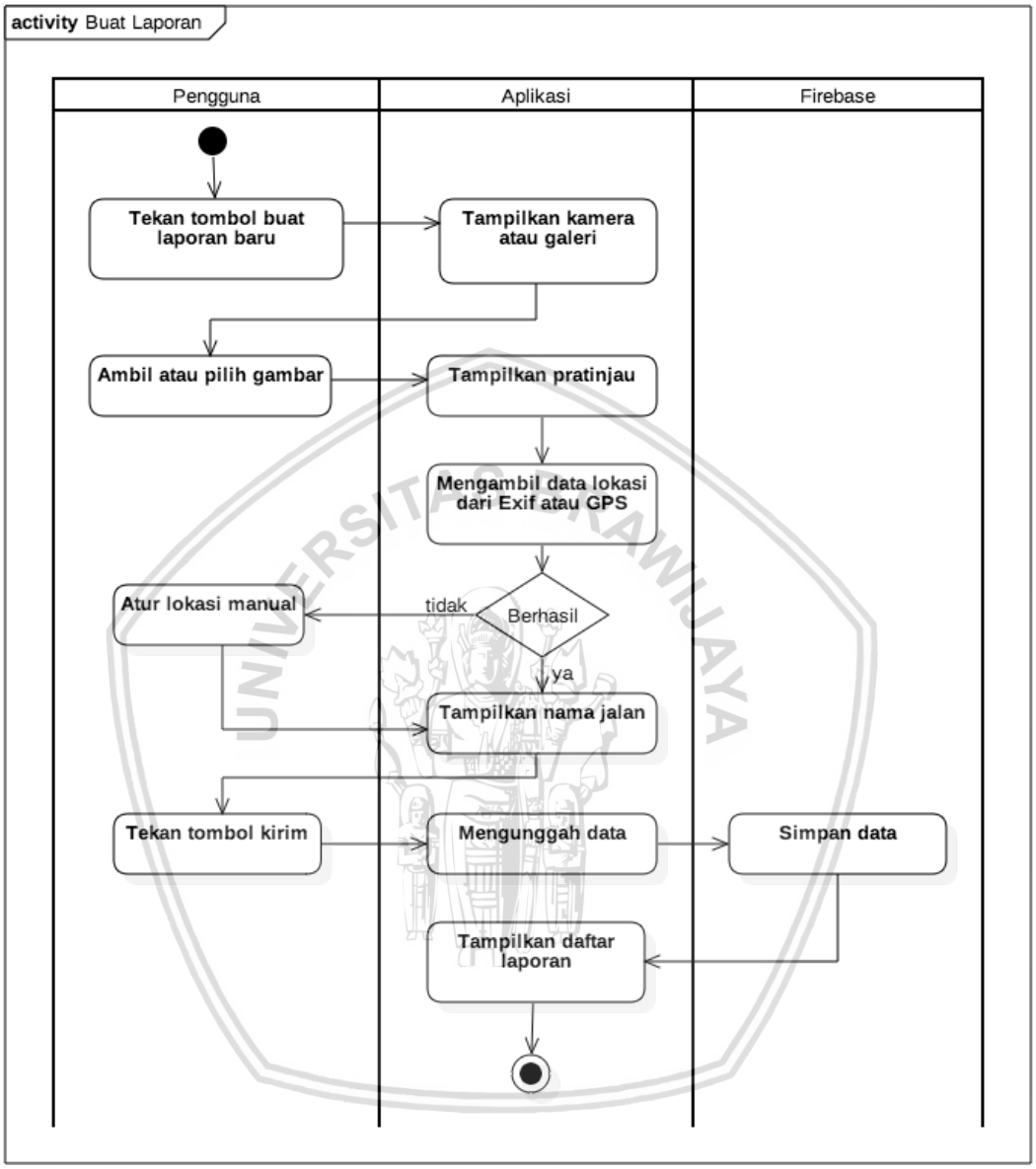
Gambar 4.5 Perancangan arsitektur sistem laporan kerusakan jalan

4.2.2 Perancangan *Activity Diagram*

Activity diagram berfungsi untuk memodelkan aktifitas yang berupa interaksi antara pengguna dan sistem berdasarkan skenario *use case*. *Activity diagram* digambarkan secara urut dan dikelompokkan berdasarkan grup tertentu menjadi beberapa grup. *Activity diagram* dari sistem aplikasi ini ditunjukkan pada

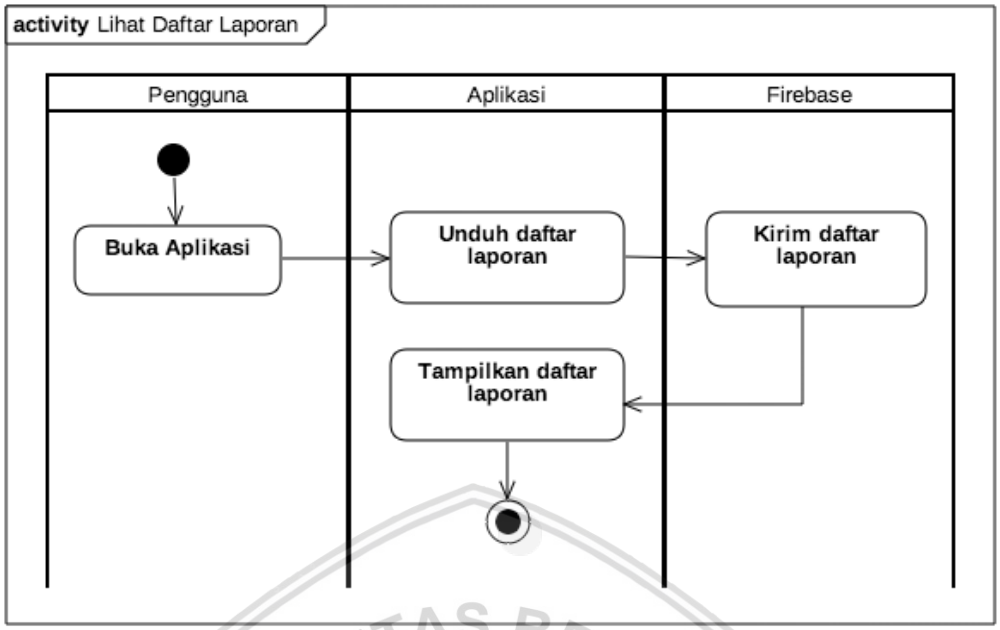
Gambar 4.6 adalah *activity diagram* dari aktifitas untuk membuat laporan baru. Pada diagram ini digambarkan bagaimana proses pembuatan laporan kerusakan jalan baru oleh pengguna aplikasi *mobile*. Pengguna dapat memilih untuk mengambil foto dengan sensor kamera atau memilih gambar pada galeri. Setelah itu akan ditampilkan pratinjau laporan baru dan aplikasi akan mencoba untuk membaca data lokasi pada foto yang dipilih, jika gagal aplikasi akan

mencoba untuk mengambil data lokasi pengguna saat ini menggunakan sensor GPS. Jika masih tetap gagal, pengguna dapat mengatur lokasi secara manual melalui tampilan peta *location picker*.

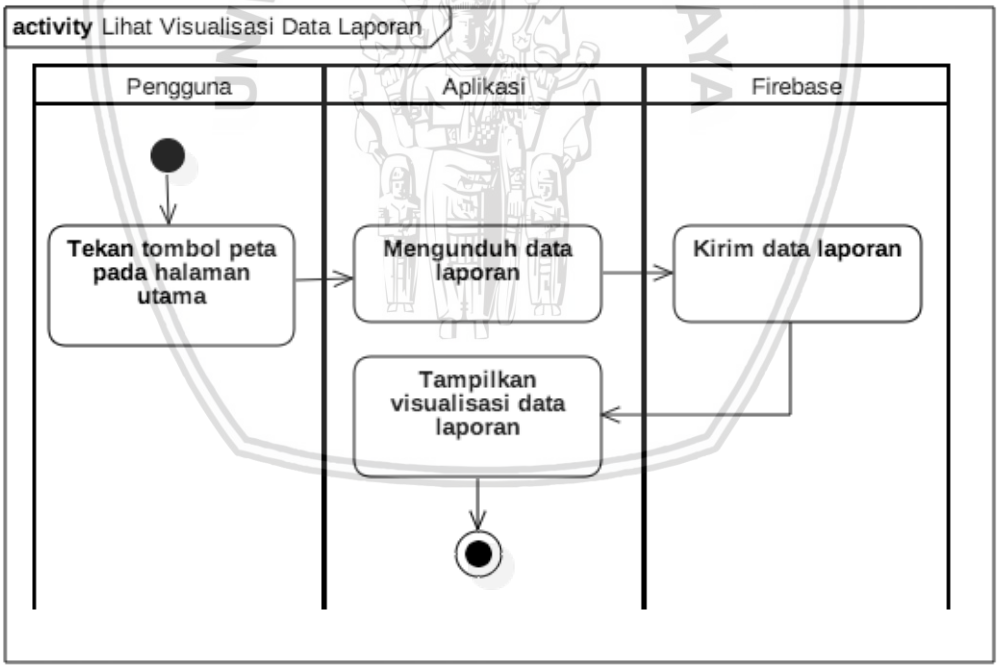


Gambar 4.6 *Activity diagram* membuat laporan baru

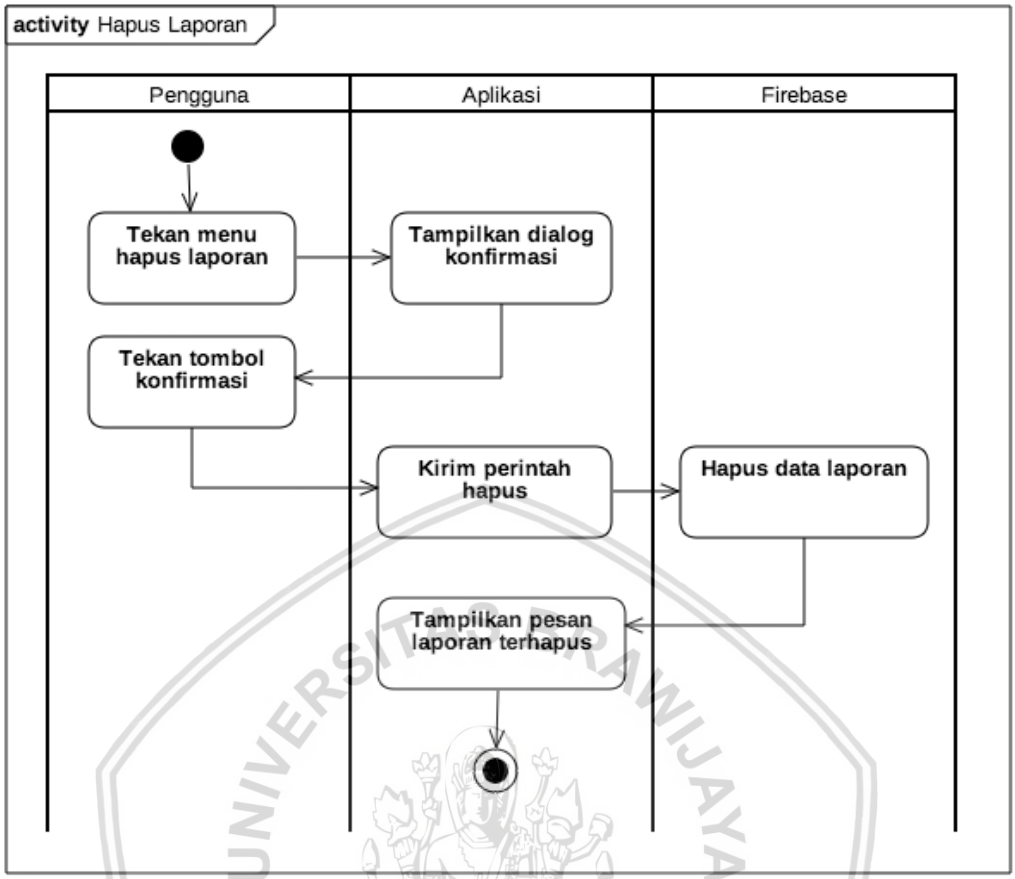
Gambar 4.7 dan Gambar 4.8 menggambarkan alur aktifitas untuk melihat daftar laporan kerusakan jalan dan tampilan visualisasi data laporan. Untuk daftar laporan akan ditampilkan dalam bentuk *timeline*. Untuk visualisasi data, ditampilkan dalam bentuk *clustered map* dan *heat map*.



Gambar 4.7 *Activity diagram* lihat daftar laporan



Gambar 4.8 *Activity diagram* lihat visualisasi data laporan

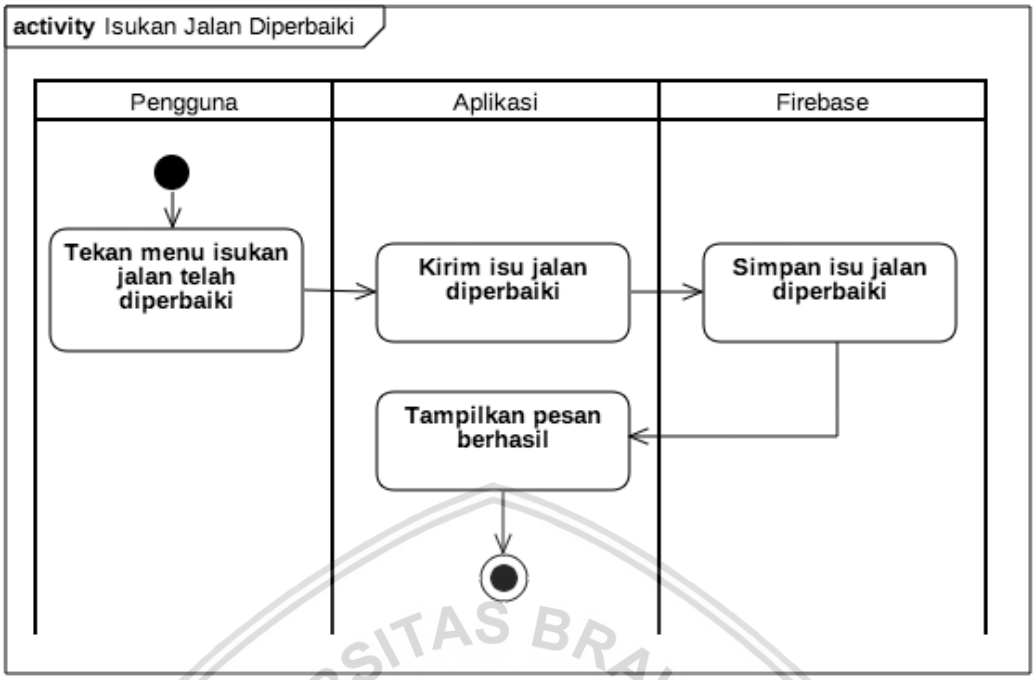


Gambar 4.9 *Activity diagram* hapus laporan

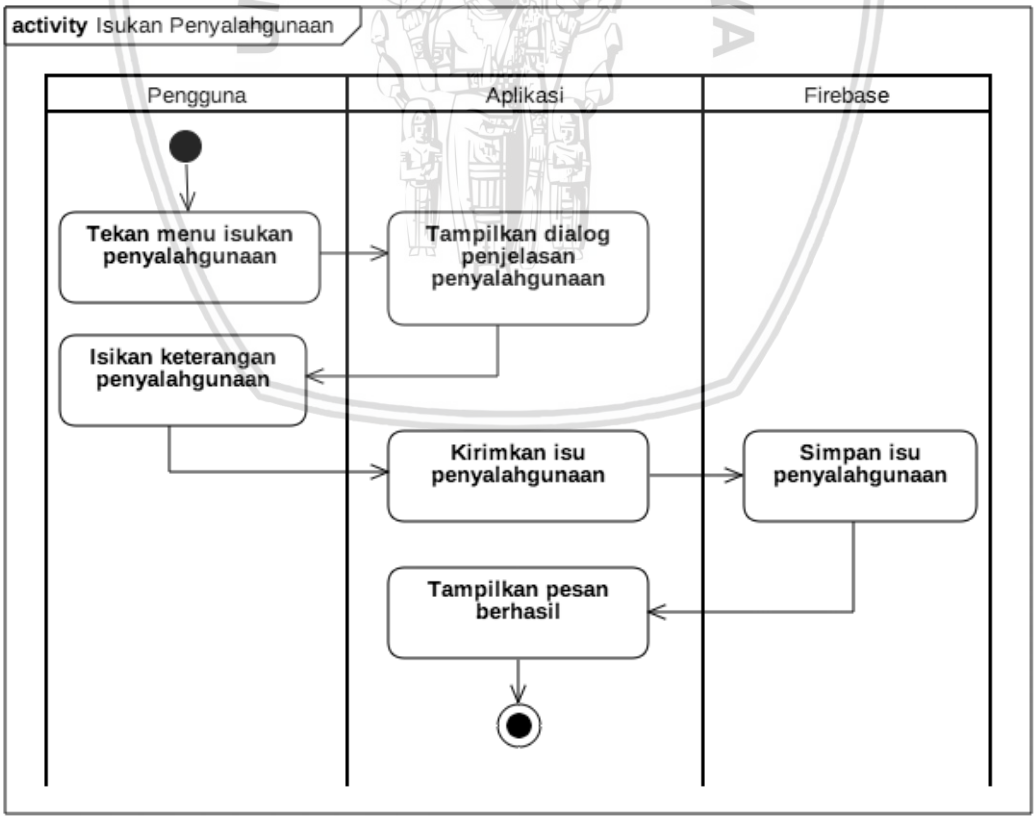
Pada Gambar 4.9, adalah alur aktifitas hapus laporan kerusakan jalan. Laporan yang dapat dihapus hanya laporan dari pengguna yang sedang terautentikasi (*login*). Setelah menekan tombol menu hapus laporan, akan ditampilkan dialog konfirmasi untuk memastikan laporan benar-benar ingin dihapus. Hal ini untuk mencegah kesalahan pengguna, sehingga laporan tidak terhapus secara tidak sengaja. Setelah laporan terhapus akan ditampilkan pesan bahwa laporan telah berhasil dihapus pada tampilan *snackbar*.

Selain dapat melaporkan kerusakan jalan, pengguna juga dapat mengisukan laporan kerusakan jalan yang telah diperbaiki dan juga penyalahgunaan seperti yang digambarkan *activity diagram* pada Gambar 4.10 dan Gambar 4.11 di bawah. Untuk isu penyalahgunaan disertakan keterangan atas penyalahgunaan yang dilakukan oleh pengguna terlapor.

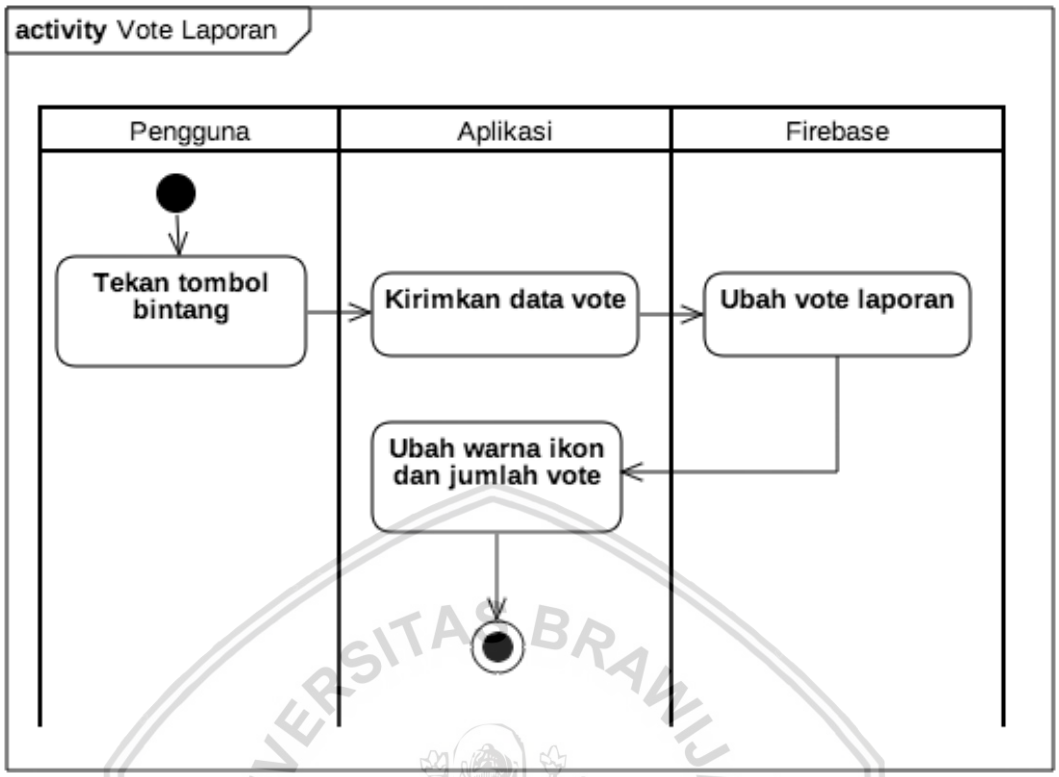
Gambar 4.12 menunjukkan alur aktifitas *vote* laporan. Pengguna hanya bisa melakukan *vote* pada laporan kerusakan jalan yang dilaporkan oleh pengguna lain. Ini karena jumlah *vote* pada laporan nantinya akan menentukan pembobotan pada visualisasi data *heat map*. Jika *vote*/bobot laporan semakin tinggi, maka intensitas *heat map* juga semakin tinggi. Fitur ini berfungsi agar pengguna tidak perlu membuat laporan baru ketika menemukan jalan rusak yang telah dilaporkan oleh pengguna lain.



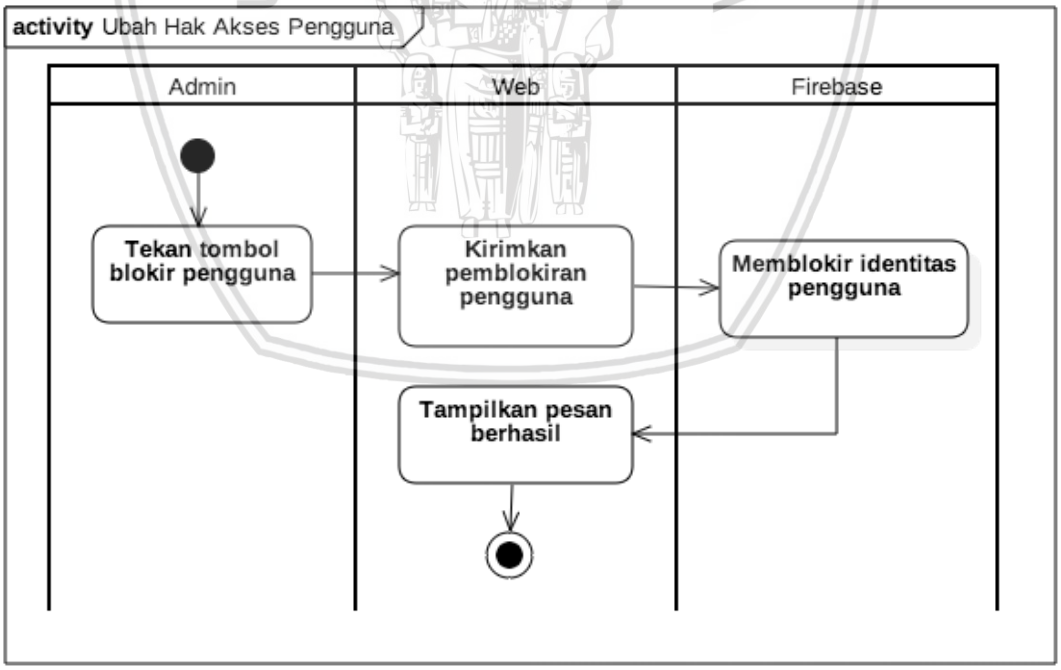
Gambar 4.10 *Activity diagram* isukan jalan diperbaiki



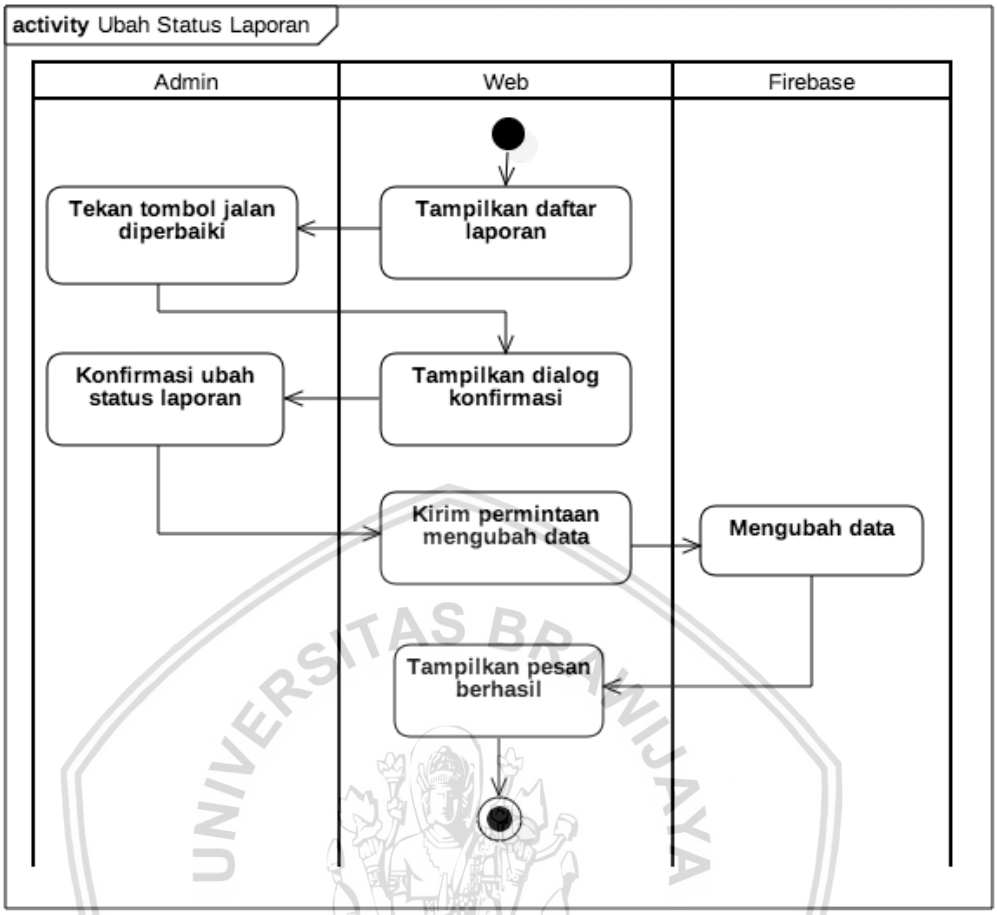
Gambar 4.11 *Activity diagram* isukan penyalahgunaan



Gambar 4.12 *Activity diagram vote* laporan



Gambar 4.13 *Activity diagram* ubah hak akses pengguna



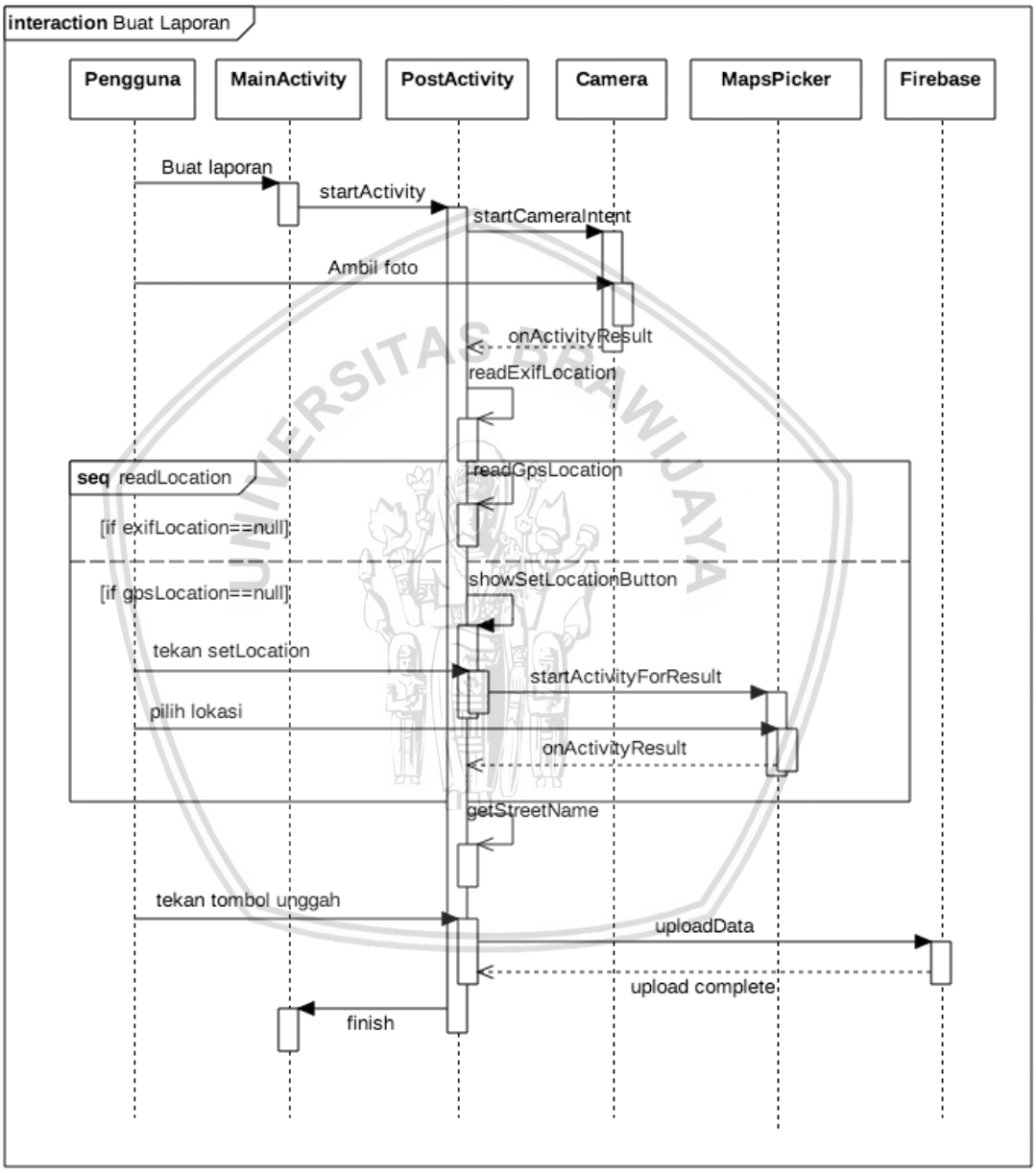
Gambar 4.14 *Activity diagram* ubah status laporan

Gambar 4.13 merupakan *activity diagram* pada aplikasi *web* untuk admin sebagai salah satu alur aktifitas dari fitur kelola pengguna. Admin dapat memblokir identitas pengguna jika dinilai pengguna telah melakukan penyalahgunaan, baik dari penilaian admin langsung maupun dari isu yang dikirimkan oleh pengguna lain. Setelah identitas pengguna diblokir, pengguna tersebut tidak dapat mengakses aplikasi *mobile* laporan kerusakan jalan.

Admin dapat mengubah status laporan kerusakan jalan jika jalan yang dilaporkan telah diperbaiki seperti yang ditunjukkan pada Gambar 4.14. Admin dapat mengubah status laporan yang telah diisukan oleh pengguna maupun yang tidak diisukan jika memang benar jalan yang dilaporkan telah diperbaiki. Aktifitas ini merupakan bagian dari fitur kelola laporan kerusakan jalan untuk admin. Selain dapat mengubah status laporan, admin juga dapat melihat daftar laporan, dan menghapus data laporan jika dinilai laporan tersebut disalahgunakan. Untuk *activity diagram* lihat dan hapus laporan pada dasarnya sama seperti *activity diagram* pada pengguna aplikasi *mobile*. Hanya saja admin dapat menghapus laporan dari semua pengguna.

4.2.3 Perancangan *Sequence Diagram*

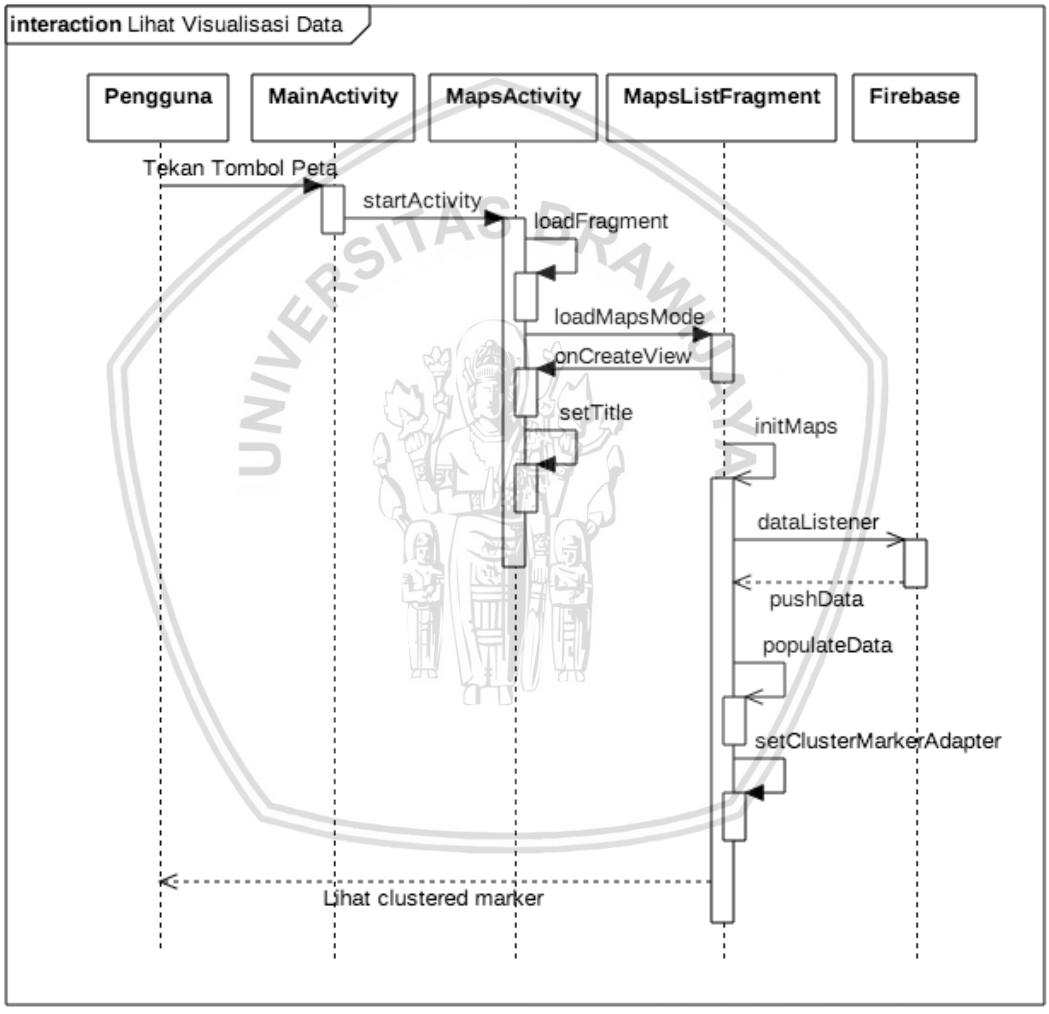
Sequence diagram memodelkan interaksi dalam suatu sistem dengan menekankan urutan waktu. *Sequence diagram* terdiri dari beberapa objek yang disusun secara horizontal. Kemudian dari objek-objek tersebut ditarik garis vertikal ke bawah yang merepresentasikan alur waktu. Dalam penelitian ini, penulis memodelkan beberapa fitur dari analisis *use case*.



Gambar 4.15 *Sequence diagram* membuat laporan baru

Gambar 4.15 menunjukkan urutan interaksi ketika pengguna aplikasi *mobile* membuat laporan kerusakan jalan baru. Pada *sequence diagram* tersebut hanya digambarkan ketika pengguna membuat laporan kerusakan jalan baru menggunakan kamera. Untuk proses membuat laporan baru menggunakan galeri prosesnya hampir sama. Dapat dilihat terdapat alur alternatif pada saat

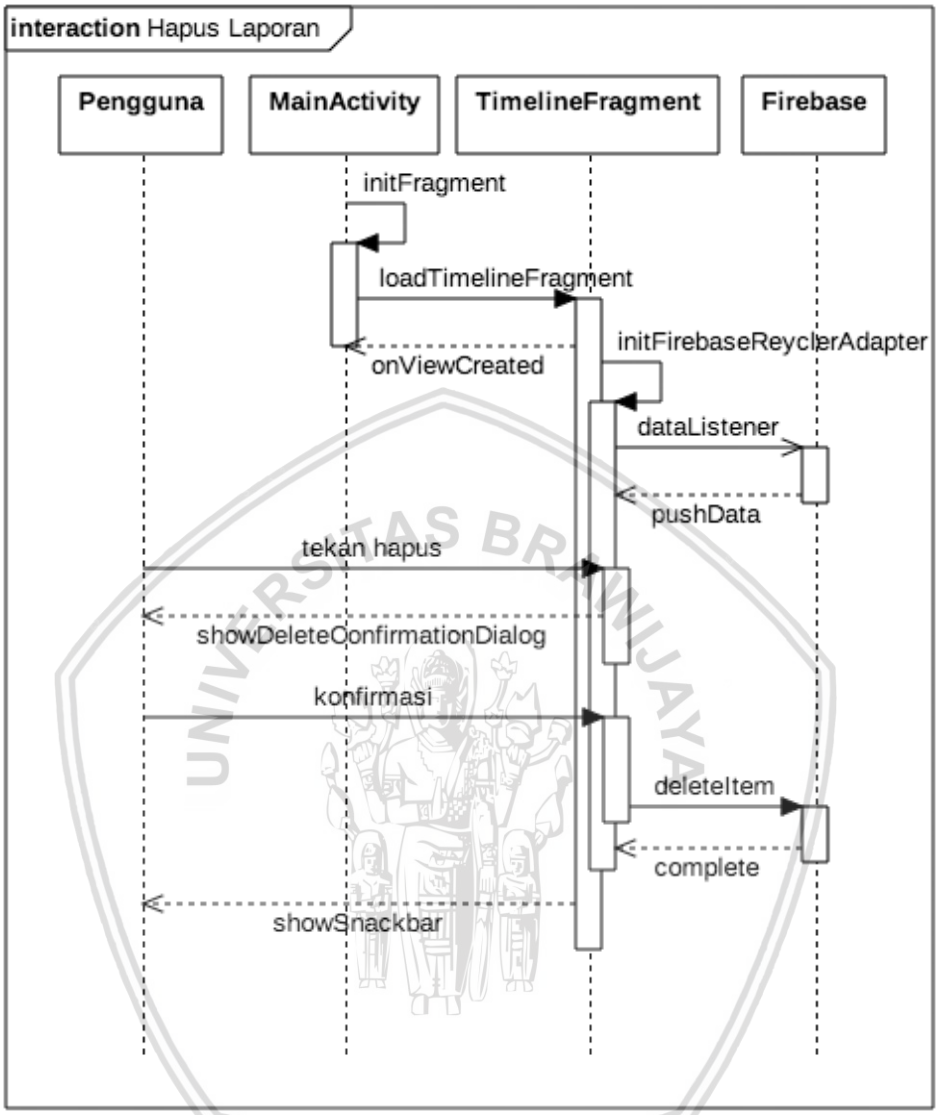
pembuatan laporan baru. Aplikasi *mobile* akan mencoba untuk membaca data lokasi pada EXIF foto terlebih dahulu. Jika data lokasi pada EXIF tidak ditemukan, makan aplikasi akan mencoba mengambil lokasi pengguna saat ini menggunakan sensor GPS. Jika pengguna menolak untuk menyalakan sensor GPS atau data lokasi tidak dapat ditemukan, maka aplikasi akan memunculkan tombol atur lokasi manual. Setelah data lintang dan bujur lokasi didapatkan, aplikasi akan mencari nama jalan terdekat pada lokasi tersebut. Jika nama jalan sudah ditemukan, tombol untuk mengunggah laporan akan dapat diakses oleh pengguna. Selain itu, pengguna juga diperbolehkan mengisi deskripsi laporan yang bersifat opsional untuk menjelaskan lebih rinci tentang kerusakan jalan yang akan dilaporkan.



Gambar 4.16 *Sequence diagram* lihat peta visualisasi data

Gambar 4.16 merupakan *sequence diagram* untuk melihat visualisasi data laporan kerusakan jalan pada aplikasi *mobile*. Ada dua mode visualisasi data, yaitu *clustered marker* dan *heat maps*. Keduanya menggunakan API yang sudah disediakan Google Maps. Untuk *clustered marker*, data dipopulasikan menggunakan ClusterMarkerAdapter. Sedangkan untuk *heat maps*, data dipopulasikan menggunakan HeatmapTileProvider. Visualisasi data *heatmaps*

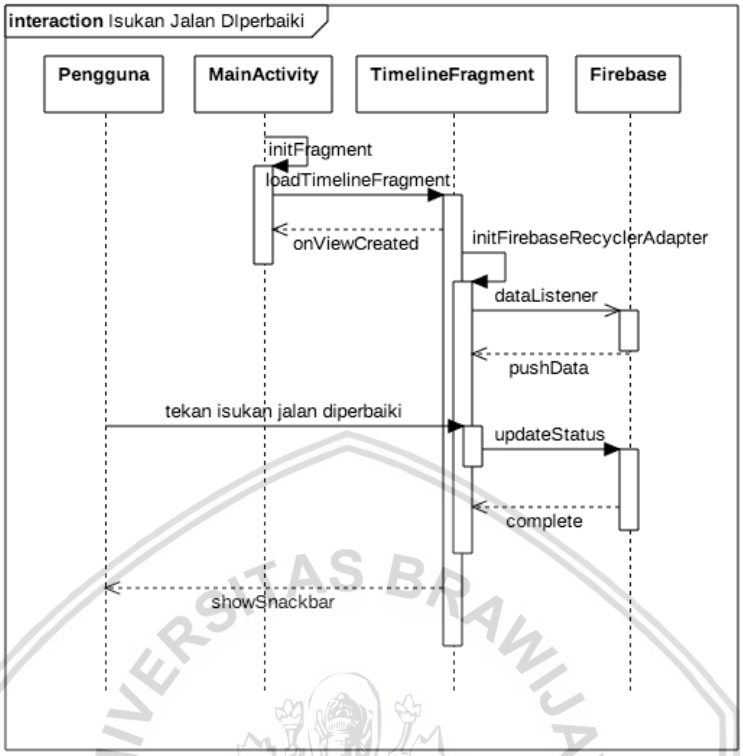
memiliki nilai intensitas berdasarkan jumlah laporan dan pembobotan nilai berdasarkan jumlah *vote* pada suatu laporan.



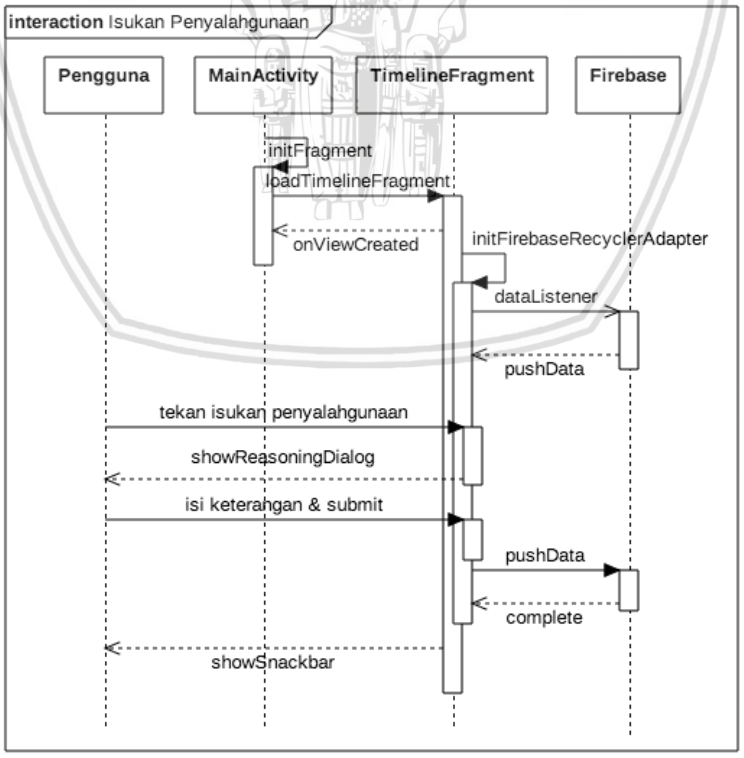
Gambar 4.17 *Sequence diagram* hapus laporan

Untuk proses interaksi penghapusan laporan, dapat dilihat pada Gambar 4.17. Ditunjukkan pada *sequence diagram* hapus laporan, pengguna harus melakukan konfirmasi penghapusan laporan untuk mengatasi kondisi ketika pengguna secara tidak sengaja menekan tombol menu hapus pada laporan.

Seperti yang sudah dibahas sebelumnya pada analisis fungsional, pengguna dapat mengisukan laporan kerusakan jalan yang telah diperbaiki. Untuk *sequence diagram* dari proses tersebut dapat dilihat pada Gambar 4.18. Selain dapat mengisukan jalan diperbaiki, pengguna aplikasi *mobile* juga dapat mengisukan penyalahgunaan laporan seperti pada Gambar 4.19. Pengguna yang melaporkan dapat mengisikan keterangan penyalahgunaan. Kemudian data akan dikirim ke server Firebase untuk dapat ditindaklanjuti oleh admin.

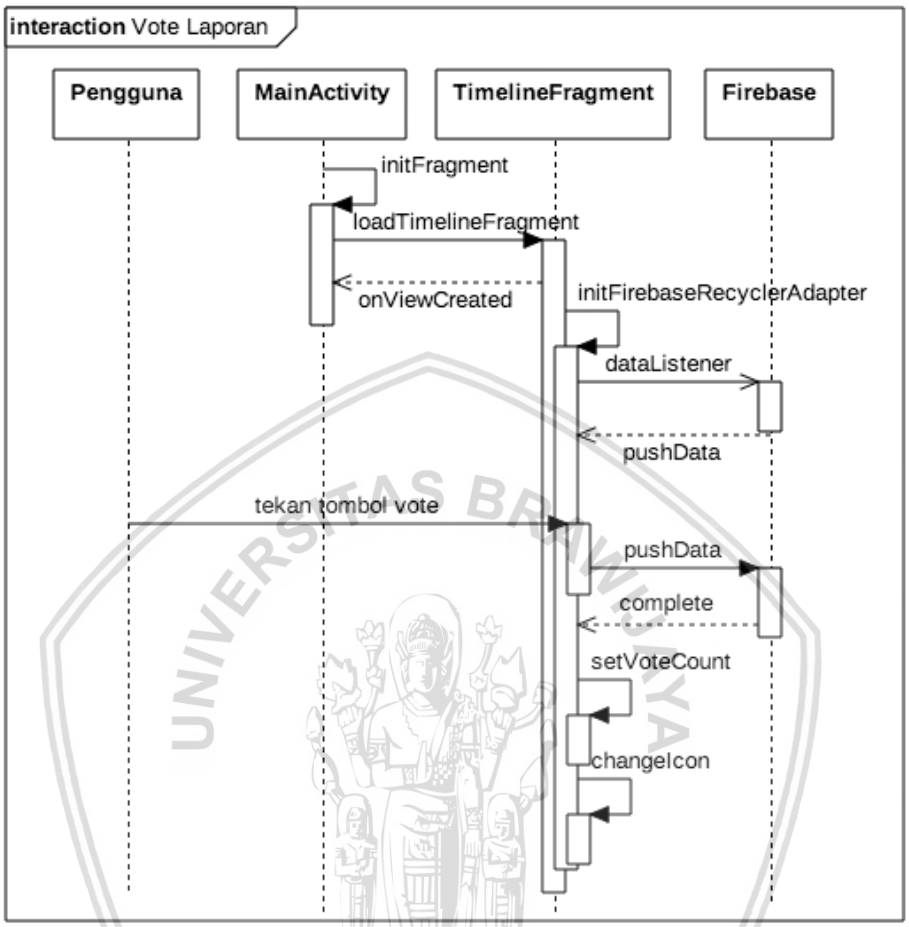


Gambar 4.18 *Sequence diagram* isukan jalan diperbaiki



Gambar 4.19 *Sequence diagram* isukan penyalahgunaan

Gambar 4.20 menunjukkan *sequence diagram* yang menggambarkan alur proses *vote* laporan. Pengguna dapat melakukan *vote* pada laporan pengguna lain untuk menaikkan bobot nilai laporan tersebut.

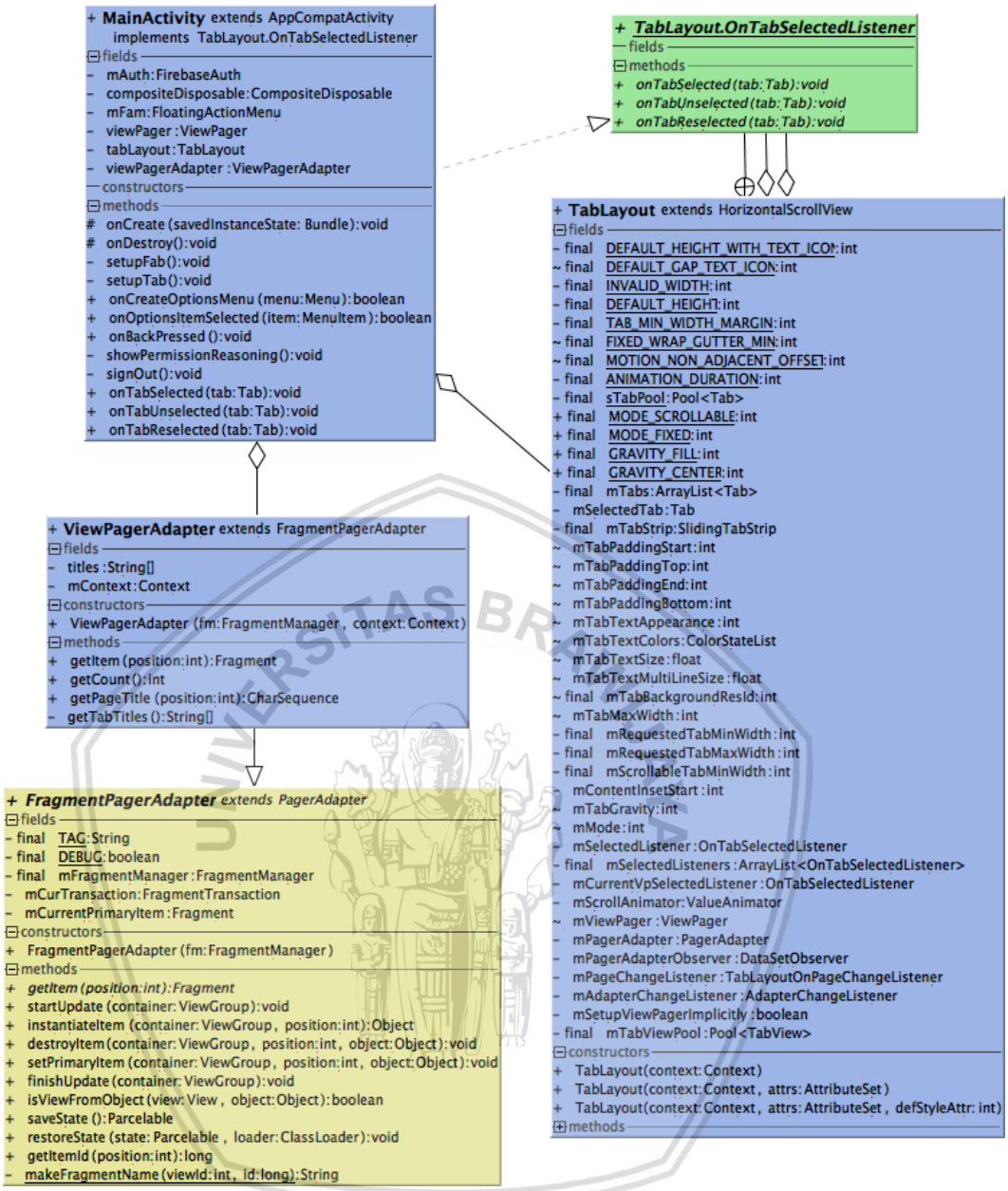


Gambar 4.20 *Sequence diagram vote* laporan

4.2.4 Perancangan *Class Diagram*

Class diagram menggambarkan *class*, *abstract class*, *interface* pada pemrograman berorientasi objek. Pada perancangan diagram ini, ditunjukkan hubungan antar objek satu dengan lainnya. Sehingga dapat diketahui bagaimana struktur rancangan dari sistem yang akan dikembangkan.

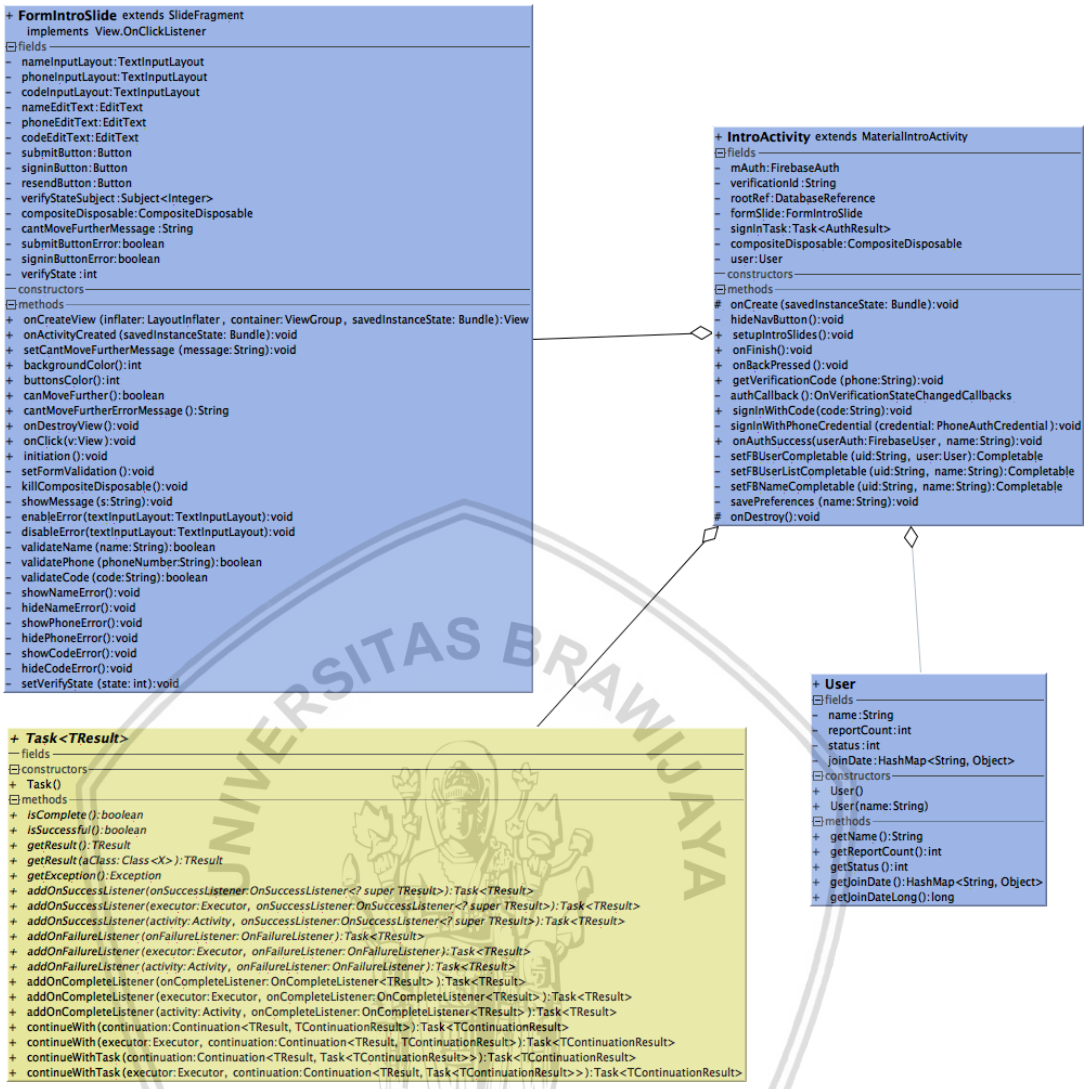
Pada Gambar 4.21 ditunjukkan kelas MainActivity dan beberapa kelas lain, diantaranya: TabLayout, ViewPagerAdapter, dan FragmentPagerAdapter. Kelas-kelas tersebut membentuk tampilan utama dari sistem aplikasi *mobile.* Data laporan kerusakan jalan ditampilkan dalam tiga tampilan berebentuk *tab*. Tiga *tab* tampilan data terdiri dari *tab* linimasa (*timeline*), *tab* jalan yang telah diperbaiki, dan *tab* laporan oleh pengguna yang sedang *login* untuk memudahkan jika pengguna ingin melihat laporannya sendiri. Seperti pada saat ingin menghapus laporan kerusakan jalan yang salah.



Gambar 4.21 *Class diagram* utama

Gambar 4.22 menunjukkan *class diagram* untuk proses yang berjalan ketika pertama kali membuka aplikasi *mobile.* Pengguna perangkat Android versi 6.0 harus memberikan izin penggunaan sensor GPS pada proses ini. Selain itu, pengguna harus melakukan *login* dengan menggunakan nomor ponsel pengguna pada proses ini untuk dapat mengakses aplikasi.

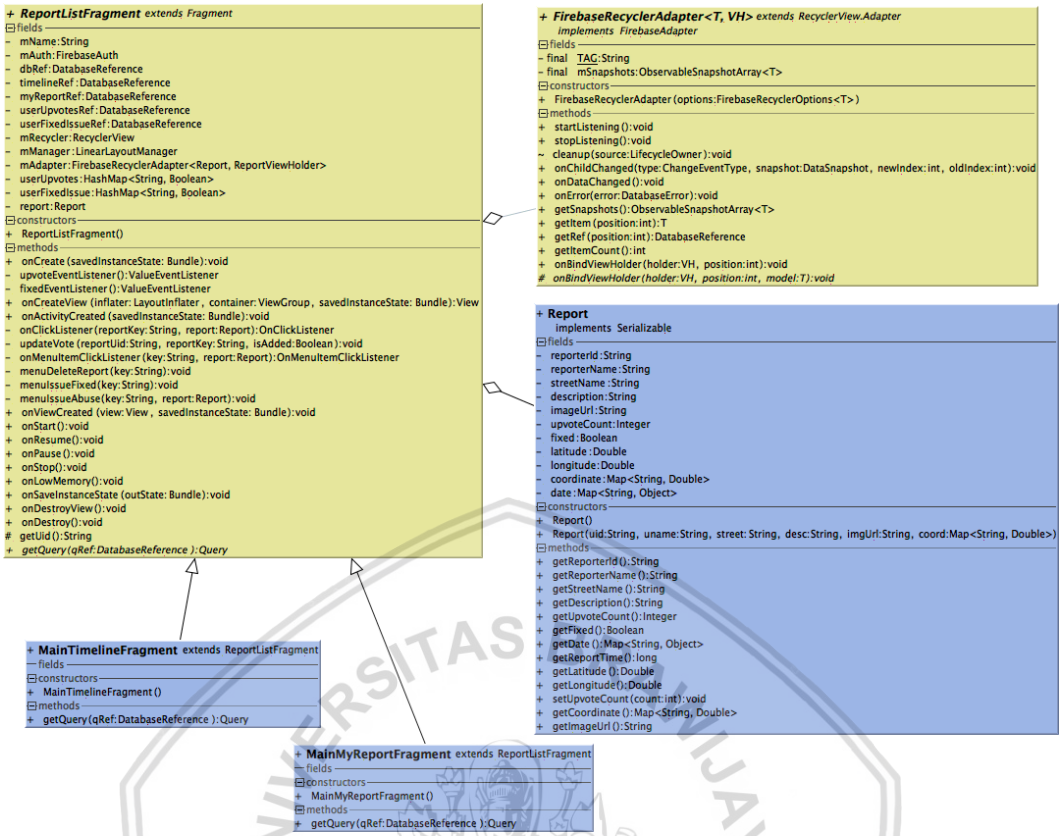
Pada Gambar 4.23 digambarkan bagaimana struktur *fragment* dari kelas utama. Dari *class diagram* tersebut dapat diketahui tampilan *tabs* pada kelas utama menggunakan tiga *fragment tampilan*, yaitu tampilan *timeline*, *fixed*, dan *my post*.



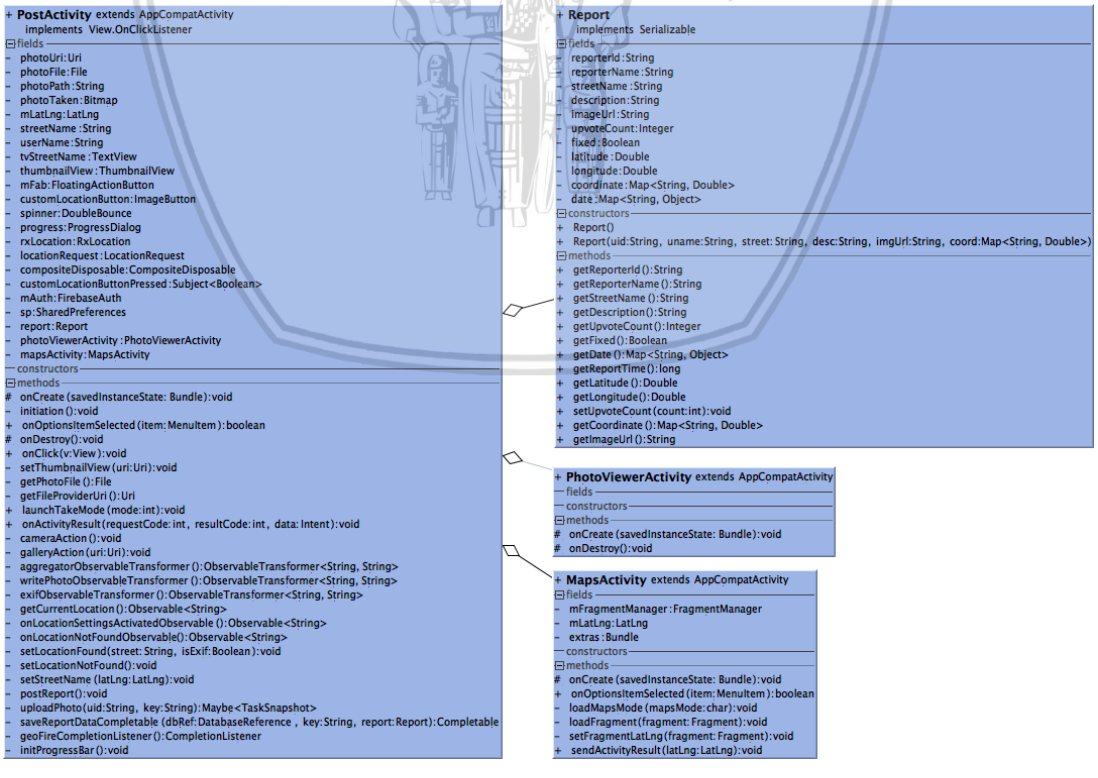
Gambar 4.22 *Class diagram intro*

Untuk Gambar 4.24 menggambarkan *class diagram post* yang berfungsi untuk membuat laporan baru. Kelas *PostActivity* dapat mengakses *MapsActivity* untuk mengatur data lokasi secara manual. Kemudian data laporan disimpan dalam bentuk struktur kelas *Report.*

Gambar 4.25 menunjukkan rancangan struktur kelas untuk fitur yang berhubungan dengan peta, seperti melihat visualisasi data dan mengatur lokasi manual pada saat membuat laporan baru. Kelas yang ada di dalamnya antara lain *MapsActivity*, *MapsListFragment*, *MapsPickerFragment*. Kelas *Report* digunakan untuk menampung struktur data yang diterima dari Firebase *realtime database*. Kedua *fragment* peta mengimplementasikan *interface OnMapReadyCallback* dan *View.OnClickListener*.



Gambar 4.23 *Class diagram* dalam *package fragment*

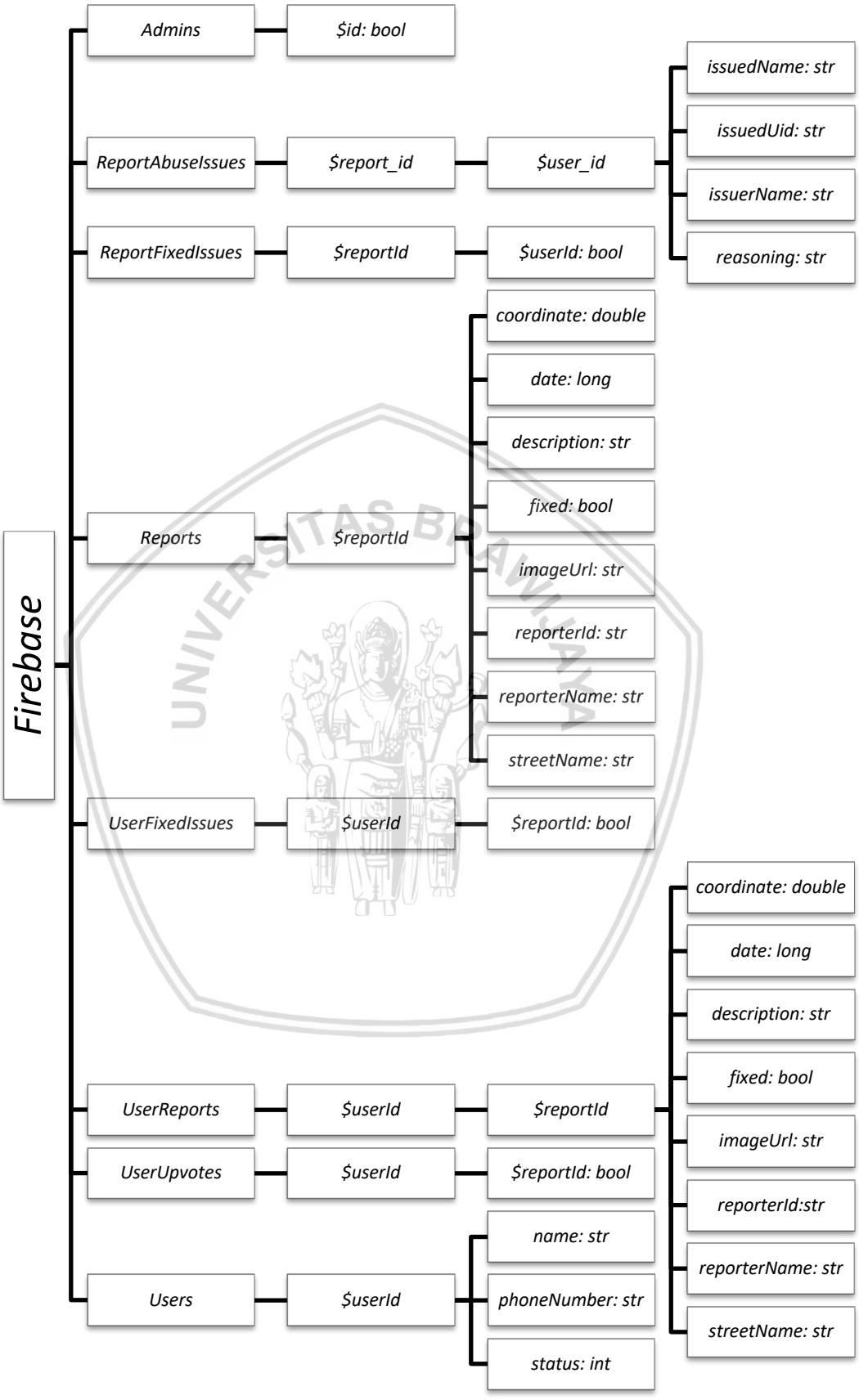


Gambar 4.24 *Class diagram* post

Gambar 4.25 *Class diagram maps*

4.2.5 Perancangan Struktur Data

Perancangan struktur data berfungsi untuk menggambarkan struktur penyimpanan data pada suatu sistem. Data sistem aplikasi penelitian ini disimpan pada Firebase *realtime-database* dalam bentuk *JSON*. Sangat disarankan untuk membuat struktur data yang ter-denormalisasi pada Firebase *realtime-database* yang berkebalikan dengan struktur *database* pada umumnya. Hal ini dikarenakan prinsip dasar Firebase *realtime-database* yang lebih mengutamakan waktu ketersediaan data secepat mungkin, daripada struktur data yang ringkas dan tidak berulang-ulang.



Gambar 4.26 Struktur data Firebase *Realtime Database*

Firebase *realtime-database* memperbolehkan kedalaman data *JSON* hingga 32 tingkat. Ketika suatu *node* data dibaca dari *server* Firebase, semua data yang berada pada tingkat di bawahnya akan ikut terbaca. Seperti halnya peraturan keamanan akses data pada Firebase *realtime-database*, setiap *node* data akan mengikuti aturan keamanan yang tertulis pada *node* tertinggi. Sehingga, sangat disarankan untuk merancang struktur data sedangkal mungkin. Selain itu, struktur data pada Firebase *realtime-database* baiknya dirancang sesuai dengan struktur tampilan aplikasi. Dalam penelitian ini, aplikasi memiliki fitur melihat laporan, data laporan untuk fitur ini dibuat pada satu node khusus. Begitu juga pada fitur melihat visualisasi data laporan kerusakan jalan, dan fitur-fitur yang lain.

Pada struktur data sistem aplikasi penelitian ini, terdapat 8 *node* dasar Firebase *realtime-database*, antara lain: *Admins, ReportAbuseIssues, ReportFixedIssues, Reports, UserFixedIssues, UserReports, UserUpvotes,* dan *Users*. *Node Admins* digunakan untuk menyimpan data identitas admin sistem. *Node ReportAbuseIssues* digunakan untuk menyimpan isu penyalahgunaan laporan. *ReportFixedIssues* dan *UserFixedIssues* digunakan untuk menyimpan data isu jalan diperbaiki dari dua sisi untuk memudahkan proses pembacaan data, yakni dari sisi data laporan dan data pengguna. *Node Reports* merupakan *node* utama untuk menyimpan laporan kerusakan jalan, sedangkan *UserReports* merupakan *node* laporan kerusakan jalan dari sisi pengguna untuk memudahkan proses pengambilan data sesuai dengan panduan struktur data pada dokumentasi Firebase. *UserUpvotes* digunakan untuk menyimpan data *vote* oleh pengguna. *Users* untuk menyimpan data pengguna terdaftar.

Terdapat duplikasi data laporan pada struktur data, hal ini untuk memudahkan pengambilan data laporan yang disesuaikan dengan tampilan aplikasi. Dalam sistem aplikasi ini data laporan ditampilkan dalam tiga bentuk, yaitu lini masa, laporan kerusakan jalan yang telah diperbaiki, dan laporan oleh pengguna yang sedang *login*. Data laporan pada *node UserReports* diambil ketika menampilkan data laporan dari pengguna yang sedang *login*. Sedangkan *node Reports* diambil ketika menampilkan data laporan kedalam bentuk lini masa dan laporan kerusakan jalan yang telah diperbaiki. Rancangan struktur data sistem dapat dilihat pada Gambar 4.26.

4.2.6 Perancangan Navigasi Antarmuka

Suatu aplikasi sebaiknya tidak mempersulit pengguna dalam mengoperasikannya. Untuk itu diperlukan perancangan navigasi dan antarmuka yang ringkas dan fungsional. Perancangan navigasi dan antarmuka akan dibahas pada sub-bab ini. Perancangan navigasi dan antarmuka terdiri dari 2 bagian, yaitu perancangan *screen flow* yang menggambarkan alur navigasi aplikasi secara garis besar. Selanjutnya perancangan antarmuka yang menggambarkan bentuk tampilan aplikasi beserta tombol-tombol untuk menavigasikannya.

4.2.6.1 Perancangan *Screen Flow*

Perancangan *screen flow* digunakan untuk menggambarkan alur antarmuka aplikasi *geotagging* laporan kerusakan jalan pada perangkat bergerak *smartphone*. Pada sistem aplikasi ini, *screen flow* dirancang se-dangkal mungkin agar tidak menyulitkan pengguna dalam mengoperasikan aplikasi sesuai dengan tujuan utama penelitian ini. Sebagian besar menu navigasi berada pada tampilan utama, sehingga untuk mengakses fitur-fitur tertentu dari aplikasi tidak membutuhkan langkah yang panjang dan rumit.

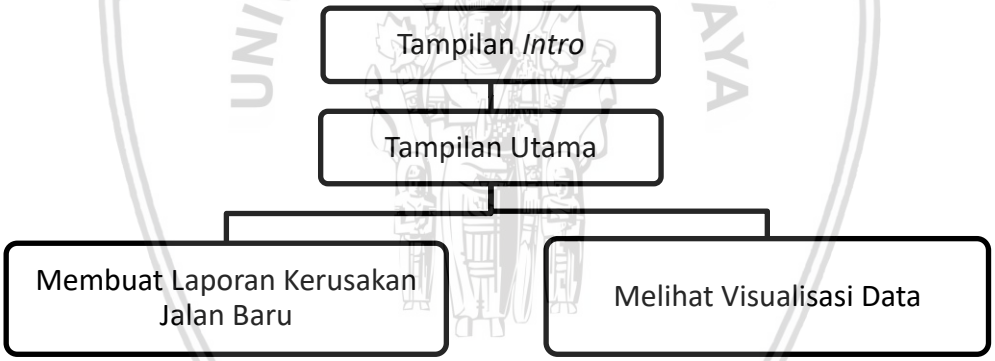
Ketika aplikasi perangkat bergerak baru dibuka, aplikasi akan melakukan pengecekan apakah pengguna sudah melakukan *login* atau belum. Jika pengguna belum melakukan *login*, aplikasi akan mengarah kan ke tampilan intro. Jika pengguna sudah melakukan *login*, akan diarahkan langsung ke tampilan utama. Di dalam tampilan intro ini berbentuk *slide* yang dapat digeser ke kanan. Terdapat 3 *slide* pada tampilan ini, *slide* yang pertama menampilkan logo aplikasi dan deskripsi singkat dari aplikasi ini. *Slide* kedua berisi penjelasan bahwa aplikasi ini membutuhkan izin untuk menggunakan sensor lokasi (dalam hal ini GPS) pada perangkat bergerak *smartphone* pengguna, pada bagian bawah terdapat tombol untuk meminta izin penggunaan sensor lokasi, jika pengguna tidak mengizinkan penggunaan sensor lokasi, pengguna tidak dapat mengakses *slide* selanjutnya. Setelah pengguna memberikan izin dengan menekan tombol pada *slide* kedua, pengguna dapat mengakses *slide* ketiga yang berisi formulir isian nama dan nomor telepon pengguna untuk melakukan proses verifikasi identitas pengguna. Setelah pengguna menekan tombol verifikasi, akan dikirimkan pesan singkat pada nomor yang dimasukkan, kemudian muncul kotak isian baru untuk kode verifikasi yang dikirimkan melalui pesan singkat tadi. Setelah pengguna mengisikan kode verifikasi, pengguna dapat masuk kedalam tampilan utama aplikasi dengan menekan tombol *Sign In*.

Dalam tampilan utama, aplikasi menampilkan daftar data laporan kerusakan jalan. Ada beberapa bentuk tampilan data yang dibagi menjadi 3 *tab*, yang pertama berisi keseluruhan data laporan kerusakan jalan dalam bentuk lini masa sesuai urutan waktu. Kedua berisi laporan kerusakan jalan yang telah diisukan diperbaiki dan telah dikonfirmasi oleh admin. Ketga berisi laporan pengguna yang sedang login untuk memudahkan pengguna melihat laporannya sendiri. Pengguna juga dapat berinteraksi dengan laporan-laporan pada tampilan ini, seperti memberikan *vote* untuk menaikkan nilai prioritas laporan, mengisukan penyalahgunaan jika laporan, juga mengisukan jalan diperbaiki. Pengguna dapat menghapus laporannya sendiri.

Tampilan berikutnya adalah pembuatan laporan baru. Pengguna dapat membuat laporan baru dengan menekan tombol *floating action button* pada tampilan utama. Selanjutnya akan ditampilkan dua tombol *floating action button* baru untuk memilih pengambilan gambar dari kamera perangkat *smartphone* atau dari galeri gambar yang telah diambil sebelumnya. Setelah mengambil atau memilih gambar, aplikasi menampilkan pratinjau laporan untuk meninjau laporan dan mengisikan deskripsi yang bersifat opsional jika memang deiperlukan. Jika

gambar yang diambil atau dipilih memiliki *meta data* lokasi, maka aplikasi akan secara otomatis mengambil data lokasi pada gambar. Jika tidak ditemukan data lokasi pada gambar, maka aplikasi akan mengambil lokasi menggunakan sensor GPS. Jika lokasi dari sensor GPS tidak ditemukan, maka pengguna dapat mengatur lokasi secara manual. Pengguna dapat selalu mengubah lokasi secara manual untuk melakukan koreksi jika data lokasi dari sensor GPS atau EXIF gambar tidak akurat. Pengguna hanya dapat mengunggah data laporan jika data lokasi sudah diisikan dengan menekan tombol *floating action button* pada tampilan pratinjau laporan, jika data lokasi belum diisikan, tombol tersebut tidak ditampilkan.

Pengguna dapat mengakses tampilan visualisasi data laporan kerusakan jalan dalam bentuk peta lokasi. Ada 2 macam visualisasi data, yaitu *clustered-marker* dan *heatmap*. Tampilan visualisasi *clustered-marker* menandai lokasi laporan kerusakan jalan dengan *marker* yang dapat ditekan untuk melihat detail laporan. Jika terdapat banyak data laporan yang berdekatan, *marker* akan digabungkan dalam bentuk *cluster*. Sedangkan tampilan *heatmap* memvisualisasikan data laporan dalam gradien warna. Intensitas warna pada visualisasi *heatmap* berdasarkan jumlah laporan kerusakan jalan dan banyaknya *vote* pada laporan tersebut. Berikut diagram perancangan *screenflow* pada Gambar 4.27**.**

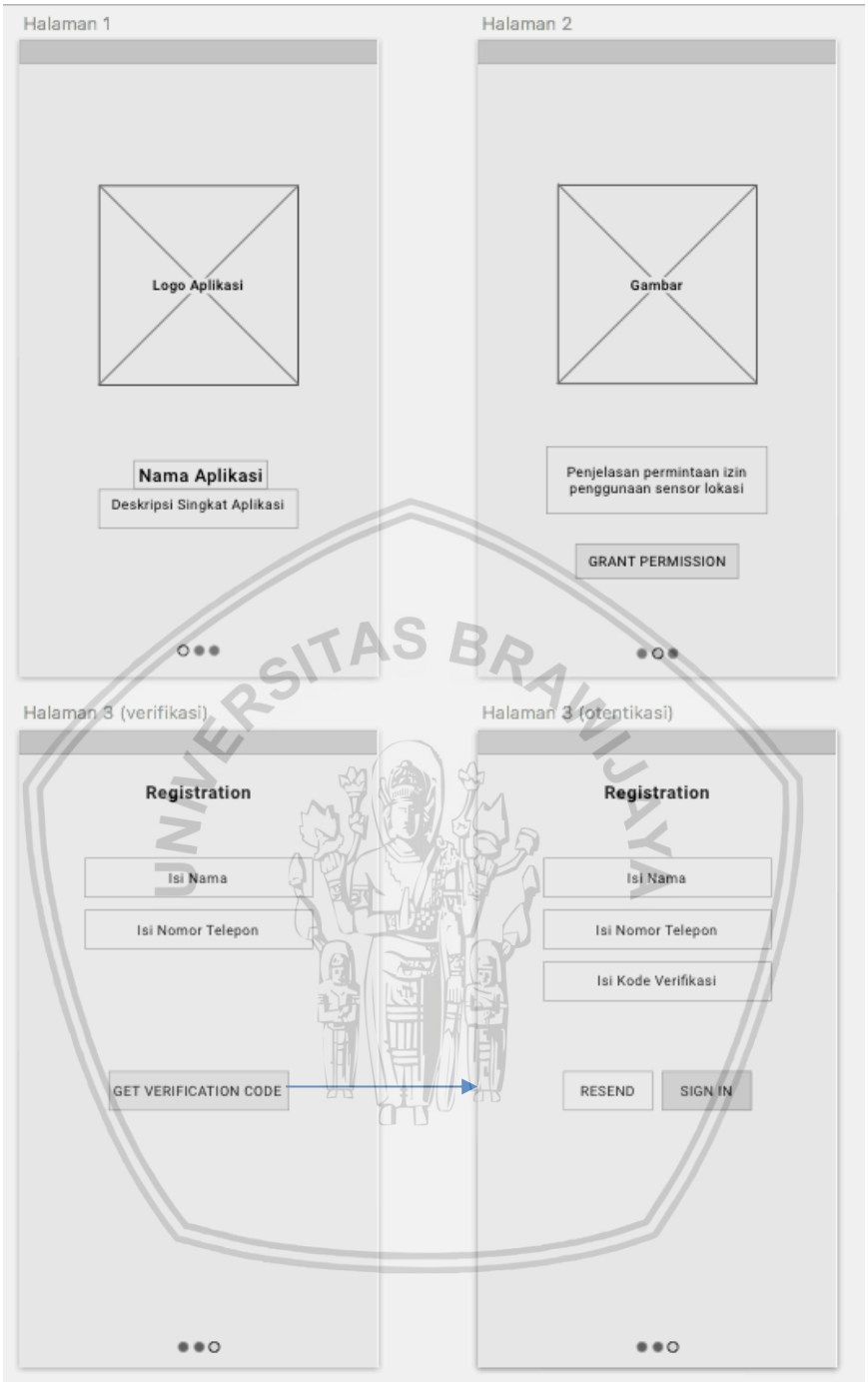


Gambar 4.27 *Screen Flow* aplikasi perangkat bergerak

4.2.6.2 Perancangan Antarmuka

1. Antarmuka Tampilan *Intro*

Antarmuka *intro* hanya ditampilkan ketika pengguna belum melakukan proses *login*, seperti ketika aplikasi baru dipasang di perangkat *smartphone* atau ketika pengguna baru saja melakukan *logout*. Halaman antarmuka *intro* dibagi menjadi tiga halaman *slide*, halaman pertama menampilkan logo, nama aplikasi, dan deskripsi singkat. Halaman kedua menampilkan penjelasan bahwa aplikasi membutuhkan izin penggunaan sensor GPS dan tombol untuk memberikan izin pada aplikasi. Yang terakhir menampilkan formulir untuk melakukan *login* menggunakan nomor telepon pengguna. Rancangan antarmuka halaman *intro* dapat dilihat pada Gambar 4.28.



Gambar 4.28 Rancangan antarmuka tampilan *intro*

2. Antarmuka Tampilan Utama

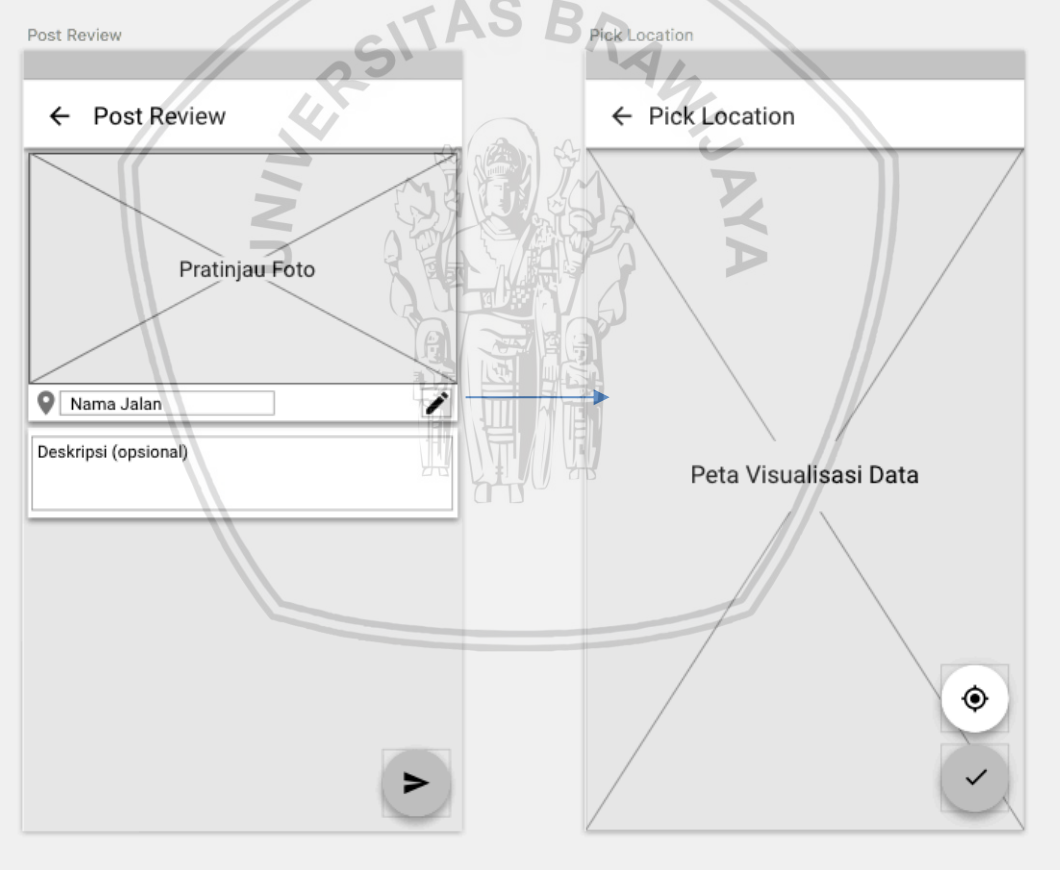
Pada tampilan untama menampilkan data laporan kerusakan jalan dan tombol untuk pengguna melakukan interaksi dengan data laporan tersebut. Seperti memberikan *vote*, mengisukan jika dalan sudah diperbaiki, mengisukan laporan yang disalahgunakan, dan menghapus laporan milik pengguna sendiri. Pengguna juga dapat membuat laporan baru dengan menekan *floating action button* pada halaman ini. Di halaman ini juga pengguna dapat melakukan navigasi ke halaman

visualisasi data laporan dengan menekan tombol pada *action bar*. Berikut rancangan tampilan utama yang ditunjukkan pada Gambar 4.29.

Gambar 4.29 Rancangan antarmuka tampilan utama

3. Antarmuka Tampilan Pratinjau Laporan Baru

Antarmuka tampilan ini digunakan untuk meninjau dan mengisi data tambahan pada laporan yang akan diunggah. Aplikasi menampilkan tampilan ini ketika pengguna telah menekan tombol *floating action button* pada tampilan utama dan telah memilih atau mengambil foto jalan rusak yang akan diunggah. Pada tampilan ini terdapat pratinjau foto yang dipilih atau diambil, pengguna dapat melihat foto tersebut untuk memastikan data yang akan dilaporkan. Selain itu terdapat nama jalan tempat foto tersebut diambil. Jika jalan yang ditampilkan tidak akurat, pengguna dapat mengatur lokasi secara manual dengan menekan tombol edit di sebelah kanan nama jalan yang memiliki ikon pensil. Selain itu pengguna juga dapat menambahkan deskripsi kerusakan jalan yang bersifat opsional pada kotak isian dibawahnya. Pengguna dapat mengunggah data dengan menekan tombol *floating action button* di pojok kanan bawah pada tampilan ini yang hanya muncul jika data lokasi telah ditemukan atau diisikan manual. Rancangan antarmuka pada tampilan ini dapat dilihat pada Gambar 4.30.

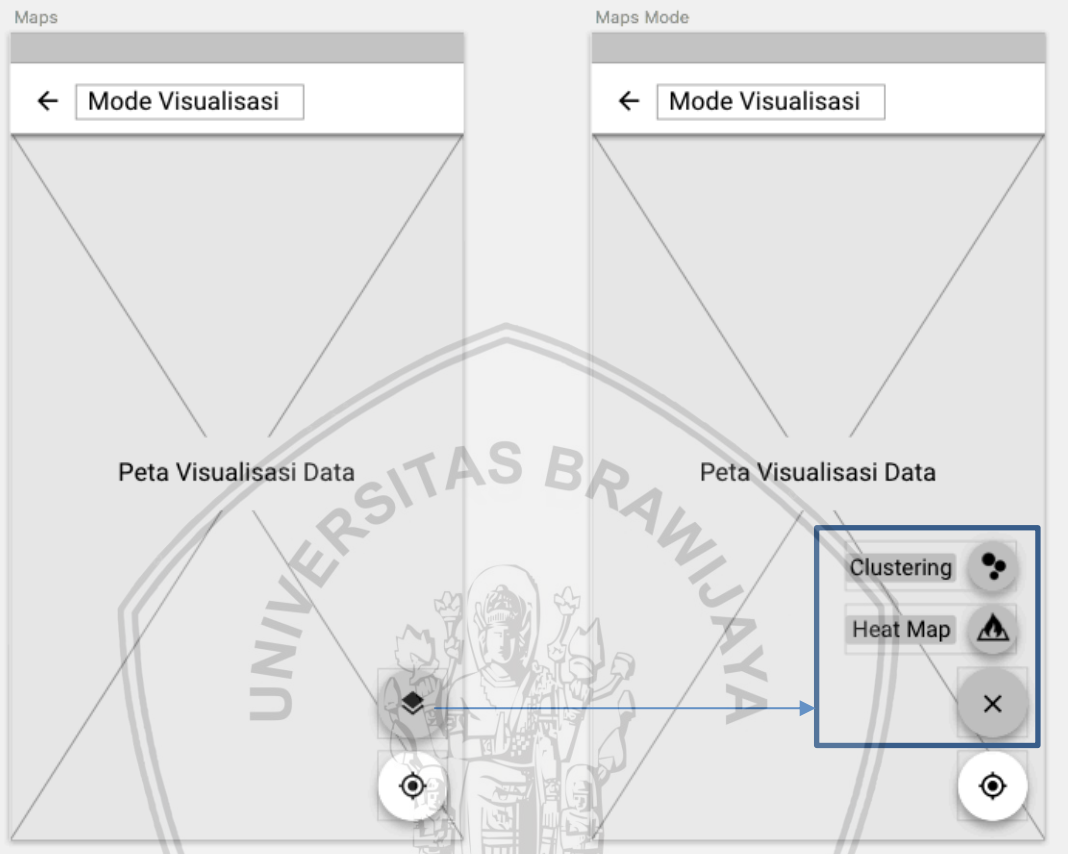


Gambar 4.30 Rancangan antarmuka pratinjau laporan baru

4. Antarmuka Lihat Peta Visualisasi Data

Pada antarmuka lihat visualisasi data, ditampilkan peta yang di dalamnya terdapat penanda di mana lokasi laporan kerusakan jalan yang telah dilaporkan. Tampilan ini dapat diakses dengan menekan ikon peta pada *action bar* tampilan utama. Terdapat dua mode visualisasi data yaitu *clustered marker* dan *heatmap*.

Mode visualisasi data dapat diganti dengan menekan *floating action button* dengan ikon *layer*, kemudian dua *sub-floating action button* akan ditampilkan yang masing-masing memiliki label mode visualisasi data. Rancangan antarmuka lihat visualisasi data ditunjukkan pada Gambar 4.31.



Gambar 4.31 Rancangan antarmuka lihat visualisasi data

BAB 5 IMPLEMENTASI

Bab ini berisi tentang bagaimana analisis dan rancangan aplikasi laporan kerusakan jalan diimplementasikan. Dalam bab ini akan dilakukan pembahasan mengenai implementasi spesifikasi lingkungan, batasan, kode program, dan antarmuka dari sistem aplikasi.

5.1 Spesifikasi Sistem

Pengembangan sistem aplikasi membutuhkan lingkungan perangkat lunak dan perangkat keras. Aplikasi penelitian ini diimplementasikan pada perangkat bergerak (*smartphone*) Android untuk sisi pengguna sebagai pelapor jalan rusak, perangkat keras yang dapat mengakses web untuk menjalankan aplikasi pada sisi admin, dan perangkat keras komputer untuk proses pengembangannya.

5.1.1 Spesifikasi Perangkat Keras

Sistem aplikasi laporan kerusakan jalan dikembangkan menggunakan komputer dengan spesifikasi yang tercantum pada Tabel 5.1.

Tabel 5.1 Spesifikasi perangkat keras komputer

Nama Komponen	Spesifikasi
Model Sistem	Macbook Pro 15” (MD103)
Prosesor	Intel® Core™ i7 (I7-3615QM) @2.3Ghz
Memori (HDD)	2 TB
Memori (RAM)	8 GB DDR3
Kartu Grafik	NVIDIA Geforce 650M

Aplikasi *client* laporan kerusakan jalan menggunakan perangkat *smartphone* Android dengan spesifikasi yang dicantumkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi perangkat keras *smartphone* Android

No	Nama Komponen	Spesifikasi
1	Model Sistem	Samsung Galaxy S8 Plus
	Prosesor	Qualcomm MSM8998 Snapdragon 835 Octa-core (4x2.35 GHz Kryo, 4x1.9 GHz Kryo)
	Penyimpanan Internal	64 GB
	Memori (RAM)	4 GB
	Layar	6.2 Inchi, 1440 x 2960, Super AMOLED
	Kamera	12 MP (f/1.7, 26mm, 1/2.5", 1.4 µm, Dual Pixel PDAF)

Tabel 5.2 Spesifikasi perangkat keras *smartphone* Android (lanjutan)

No	Nama Komponen	Spesifikasi
	WLAN	802.11 a/b/g/n/ac
	GPS	A-GPS, GLONASS, BDS, GALILEO
2	Model Sistem	Samsung Galaxy Note 3
	Prosesor	Qualcomm Snapdragon 800 Quad-core 2.3 GHz Krait 400
	Memori Internal	32 GB
	Memori (RAM)	3 GB
	Layar	5.7 Inchi, 1080 x 1920, Super AMOLED
	Kamera	13 MP (f/2.2, 31mm, 1/3", 1.12 µm)
	WLAN	802.11 a/b/g/n/ac
	GPS	A-GPS, GLONASS

5.1.2 Spesifikasi Perangkat Lunak

Sistem aplikasi laporan *geotagging* kerusakan jalan menggunakan pendekatan *Location Based Servic*e. Dalam pengembangannya digunakan komputer dengan spesifikasi perangkat lunak yang ditunjukkan pada Tabel 5.3.

Tabel 5.3 Spesifikasi perangkat lunak komputer

Nama Komponen	Spesifikasi
Sistem Operasi	MacOS 10.13.4 High Sierra
Bahasa Pemrograman	Java, XML, HTML, NodeJS
Software Development Kit	Java SE Development Kit 8
Programming Environment	Java Runtime Environtment 8 NodeJS v9.11.1
Android SDK	API 6.0
Editor	Android Studio Visual Studio Code

Pada Tabel 5.4 ditunjukkan spesifikasi perangkat lunak perangkat *smartphone* Android yang digunakan untuk instalasi dan pengujian penelitian ini.

Tabel 5.4 Spesifikasi perangkat lunak *smartphone* Android

No	Nama Komponen	Spesifikasi
1	Sistem Operasi	Android Versi 7.1.1 (Nougat)
2	Sistem Operasi	Lineage OS 15.1 / Android Versi 8.1 (Oreo)

5.2 Batasan Implementasi

Dalam pengimplementasian dari penilitian sistem aplikasi *geotagging* laporan kerusakan jalan terdapat beberapa batasan, antara lain:

1. Aplikasi *geotagging* laporan kerusakan jalan hanya mencakup wilayah Kota Malan.
2. Aplikasi harus terkoneksi dengan internet untuk dapat melakukan operasi menulis dan mengubah data.
3. Perangkat *smartphone* Android pengguna harus terpasang Google Play Services.
4. Aplikasi pada perangkat *smartphone* Android memerlukan nomor ponsel pengguna untuk dapat melakukan proses *login*.
5. Aplikasi pada perangkat *smartphone* Android memerlukan ijin penggunaan sensor GPS untuk dapat menyematkan data lokasi pada laporan kerusakan jalan secara otomatis.
6. Implementasi peta visualisasi data *clustered marker* dan *heatmap* menggunakan API Google maps.
7. Menggunakan layanan Firebase s*park plan* dengan batasan kuota penggunaan yang telah ditentukan.

5.3 Implementasi *Database Rule*

Untuk membatasi izin baca dan tulis data pada Firebase Realtime Database, dapat menggunakan fitur *database rule* pada *console* Firebase. *Database Rule* memiliki format JSON yang disesuaikan dengan skema *database*.

```
1  {
2    "rules": {
3      "Users" :{
4        ".read"  : "true",
5        "$userId" :{
6          ".write" : "auth.uid == $userId ||
7  root.child('Admins/' + auth.uid).val() === true"
8        }
9      },
10     "UsersList" :{
11       ".read"  : "auth != null",
12       ".write" : "auth != null"
```

```
13      },
14      "UserFixedIssues" :{
15        ".read"   :true,
16        "$userId" :{
17          ".write" : "(auth.uid == $userId &&
18  root.child('Users/' + auth.uid).child('status').val() === 1 )||
19  root.child('Admins/' + auth.uid).val() === true"
20        }
21      },
22
23      "Reports" :{
24        ".read"     : true,
25        "$reportId" :{
26          ".write" : "(auth != null &&
27  root.child('Users/' + auth.uid).child('status').val() === 1)||
28  root.child('Admins/' + auth.uid).val() === true"
29        }
30      },
31
32      "ReportFixedIssues" :{
33        ".read"  : "auth != null",
34        ".write" : "(auth != null &&
35  root.child('Users/' + auth.uid).child('status').val() === 1)||
36  root.child('Admins/' + auth.uid).val() === true"
37      },
38
39      "ReportAbuseIssues" :{
40        ".read"  : "auth != null",
41        ".write" : "(auth != null &&
42  root.child('Users/' + auth.uid).child('status').val() === 1) ||
43  root.child('Admins/' + auth.uid).val() === true"
44      },
45
46      "UserReports" :{
47        "$userId" :{
48          ".read"  : "true",
49          ".write" : "(auth != null &&
50  root.child('Users/' + auth.uid).child('status').val() === 1)||
```

```
51  root.child('Admins/' + auth.uid).val() === true"
52        }
53      },
54
55      "UserUpvotes" :{
56        ".read"  : "auth != null",
57        "$userId" :{
58        ".write" : "(auth.uid == $userId &&
59  root.child('Users/' + auth.uid).child('status').val() === 1) ||
60  root.child('Admins/' + auth.uid).val() === true"
61        }
62     }
63  }
```

Kode 5.1 *Database rule* Firebase RDB

Kode 5.1 *Database rule* Firebase RDB merupakan implementasi *database rule* dari Firebase RDB. Setiap *node rule* mewakili *node* database sesuai dengan rancangan skema *database*. *Database* pada Firebase RDB disusun sesuai tampilan pada aplikasi, hal ini untuk mengurangi proses *query* data pada saat proses membaca.

Node Users untuk peraturan *database* pada *node* Users, pengguna dapat membaca seluruh data untuk melihat identitas semua pengguna, tetapi pengguna tidak diperbolehkan untuk mengubah data pengguna lain. Kemudian *node* UsersList yang digunakan untuk melakukan pengecekan profil pengguna ketika melakukan proses *login*. UserFIxedIssues berisikan aturan untuk membaca dan menulis data isu jalan diperbaiki, data dapat dibaca oleh semua pengguna tetapi hanya bisa ditambahkan oleh pengguna yang sedang *login* atau oleh admin. ReportFixedIssues memiliki aturan yang sama, untuk representasi data dari sisi pengguna. ReportAbuseIssues merupakan aturan *database* untuk mengisukan penyalahgunaan. *Node* Reports berisi aturan untuk baca dan tulis data laporan kerusakan jalan, digunakan untuk merepresentasikan data pada tampilan lini masa dan tampilan data jalan telah diperbaiki. Dapat dibaca oleh semua pengguna dan hanya dapat diubah oleh pengguna yang sedang *login* atau admin. *Node* UserReports memiliki aturan yang sama, hanya saja *node* ini digunakan untuk merepresentasikan data laporan kerusakan jalan oleh pengguna yang sedang *login* pada tampilan utama. UserUpvotes digunakan untuk menyimpan data *vote* laporan kerusakan jalan, dapat dibaca dan ditulis oleh pengguna yang sedang *login*, tetapi pengguna hanya dapat mengubah data *vote* masing-masing.

5.4 Implementasi Kode Program

Kode program diimplementasikan berdasarkan rancangan *sequence diagram* dan *class diagram* yang telah dirancang sebelumnya. Program aplikasi *geotagging* laporan kerusakan jalan pada *platform* Android dituliskan menggunakan bahasa

Java. Sedangkan aplikasi web untuk admin dituliskan menggunakan bahasa HTML, Javascript, dan CSS menggunakan *framework* VueJS. Karena aplikasi ini menggunakan Firebase RDB sebagai *backend* data, tidak diperlukan bahasa pemrograman *server*.

5.4.1 Kode Program Proses Pendaftaran dan *Login*

Untuk melakukan proses *login* dan pendaftaran pada aplikasi ini, digunakan layanan Firebase *authentication*. Tampilan *login* dan pendaftaran terdapat pada *IntroActivity* di *slide* ketiga. Potongan kode untuk proses *login* dan pendaftaran ditunjukkan pada Kode 5.2.

```
1  public class IntroActivity extends MaterialIntroActivity {
2    private FirebaseAuth mAuth;
3    private String verificationId;
4    private DatabaseReference rootRef;
5    private FormIntroSlide formSlide;
6    private Task<AuthResult> signInTask;
7    private CompositeDisposable compositeDisposable;
8
9    @Override
10   protected void onCreate(Bundle savedInstanceState) {
11     super.onCreate(savedInstanceState);
12     mAuth = FirebaseAuth.getInstance();
13     if (mAuth.getCurrentUser() != null) onFinish();
14     else {
15       rootRef = FbPersistence.getDatabase().getReference();
16       compositeDisposable = new CompositeDisposable();
17       hideNavButton();
18       setupIntroSlides();
19     }
20   }
21
22   public void getVerificationCode(String phone){
23     PhoneAuthProvider.getInstance()
24     .verifyPhoneNumber(phone, Consts.VERIFY_PHONE_TIMEOUT,
25       TimeUnit.SECONDS, this, authCallback());
26   }
27
28   private PhoneAuthProvider.OnVerificationStateChangedCallbacks
29     authCallback(){
```

```
30      return new PhoneAuthProvider
31        .OnVerificationStateChangedCallbacks() {
32
33        @Override
34        public void onVerificationCompleted(PhoneAuthCredential
35          phoneAuthCredential) {
36          EditText codeEdit =
37            findViewById(R.id.intro_code_input);
38          codeEdit.setText(phoneAuthCredential.getSmsCode());
39          signInWithPhoneCredential(phoneAuthCredential);
40        }
41
42        @Override
43        public void onVerificationFailed(FirebaseException e) {
44          formSlide.initiation();
45          if(e instanceof
46            FirebaseAuthInvalidCredentialsException){
47            showMessage(e.getMessage());
48          } else if(e instanceof FirebaseAuthException){
49            showMessage(getString(R.string
50              .fbauth_error_not_authorized));
51          } else if (e instanceof
52            FirebaseTooManyRequestsException){
53            showMessage(getString(R.string
54              .fbauth_error_quota_reached));
55          } else if (e instanceof
56            FirebaseApiNotAvailableException){
57            showMessage(getString(R.string
58            .fbauth_error_service_unavailable));
59          } else
60            showMessage(getString(R.string.fbauth_error_other));
61        }
62
63        @Override
64        public void onCodeSent(String verification,
65          PhoneAuthProvider.ForceResendingToken
66          forceResendingToken) {
67          super.onCodeSent(verification, forceResendingToken);
```

```
68          formSlide.setCantMoveFurtherMessage(
69              getString(R.string.intro_form_snackbar_error3));
70          verificationId = verification;
71        }
72
73        @Override
74        public void onCodeAutoRetrievalTimeOut(String s) {
75          super.onCodeAutoRetrievalTimeOut(s);
76        }
77      };
78    }
79
80    private void signInWithPhoneCredential(PhoneAuthCredential
81      credential){
82      mAuth.signInWithCredential(credential)
83        .addOnSuccessListener(authResult -> {
84        FirebaseUser user = authResult.getUser();
85        String name = ((EditText)
86          findViewById(R.id.intro_name_input))
87          .getText().toString().trim();
88        onAuthSuccess(user, name);
89      })
90      .addOnFailureListener(e -> {
91        showMessage(e.getMessage());
92      });
93    }
94
95    public void onAuthSuccess(FirebaseUser userAuth,String name){
96      String uid = userAuth.getUid();
97      User user = new User(name);
98      DatabaseReference userListRef = rootRef
99        .child("UsersList").child(uid);
100     Completable userRegisterCompletable = RxFirebaseDatabase
101       .observeSingleValueEvent(userListRef)
102       .subscribeOn(Schedulers.io())
103       .flatMapCompletable(snap -> {
104         if (!snap.exists())
105           return setFBUserCompletable(uid, user);
```

```
106           return setFBNameCompletable(uid, name);
107       });
108       compositeDisposable
109       .add(userRegisterCompletable.subscribe());
110     }
111
112     @Override
113     protected void onDestroy() {
114       super.onDestroy();
115       if (compositeDisposable!=null)
116         if (!compositeDisposable.isDisposed()
117           compositeDisposable.dispose();
118       if (signInTask != null) signInTask = null;
119     }
120     …
121 }
```

Kode 5.2 Kode program proses *login* dan pendaftaran

Kode 5.2 menunjukkan potongan kode untuk proses *login* dan pendaftaran menggunakan *phone authentication* API Firebase. Untuk melakukan *login* dan pendaftaran pengguna harus memasukkan nomor ponsel, setelah itu pengguna akan mendapat pesan singkat yang berisi kode verifikasi. Kode verifikasi tersebut dimasukkan pada kotak *verification code*. Berikut penjelasan lebih detail mengenai Kode 5.2:

1. Baris 22-26 merupakan *method* untuk mendapatkan kode verifikasi dengan parameter nomor ponsel pengguna. Nomor ponsel akan dikirimkan menggunakan API Firebase ke server.
2. Baris 27-78 merupakan *method* untuk mengembalikan objek *callback* yang mengimplementasikan empat *interface* API Firebase *authentication*. Kode pada *callback* akan dijalankan ketika tahapan proses autentikasi berganti. Masing-masing *interface* mewakili satu tahapan proses autentikasi.
3. Baris 33-40 mengimplementasikan *interface callback* yang dijalankan ketika nomor ponsel pengguna berhasil dikirimkan ke server. Dalam *interface* ini, ditampilkan kotak isian kode verifikasi SMS yang sebelumnya tidak terlihat. Jika perangkat *smartphone* Android pengguna terpasang Google Play Services, kode verifikasi akan diisikan secara otomatis dan dilakukan pemanggilan *method signInWithPhoneCredential* pada baris 71-84 yang selanjutnya akan mengarahkan ke tampilan utama aplikasi.
4. Baris 42-59 mengimplementasikan *interface* untuk melakukan proses ketika verifikasi data gagal. Jika ada kesalahan pada proses autentikasi nomor telepon pengguna, akan ditampilkan pesan eror.

5. Baris 63-69 mengimplementasikan *interface* API Firebase *authentication* yang akan dijalankan ketika pesan singkat kode verifikasi telah terkirim pada ponsel pengguna.
6. Baris 80-93 merupakan implementasi *interface callback* yang dijalankan ketika aplikasi gagal memasukkan kode verifikasi *login* secara otomatis, sehingga dapat ditampilkan pesan agar pengguna memasukkan kode verifikasi secara manual.
7. Baris 72-85 merupakan *method* untuk melakukan proses verifikasi setelah kode verifikasi dimasukkan. Kode verifikasi akan dikirimkan ke server Firebase. Jika kode verifikasi benar akan diarahkan ke tampilan utama aplikasi melalui *method onAuthSuccess*, jika salah akan ditampilkan pesan eror.
8. Baris 95-110 merupakan *method* untuk melakukan proses penyimpanan data ketika verifikasi berhasil. Data identifikasi pengguna akan disimpan secara lokal pada perangkat *smartphone* untuk memudahkan pengambilan data ketika diperlukan.

5.4.2 Kode Program Tampilan Utama

Tampilan utama pada aplikasi *geotagging* laporan kerusakan jalan memiliki tiga *tab* tampilan yang ada pada masing-masing *fragment*. *Tab* pertama berisi data laporan kerusakan jalan dalam bentuk lini masa yang ditampilkan sesuai urutan waktu mulai dari laporan terbaru, *tab* kedua berisi data laporan kerusakan jalan yang telah diperbaiki, *tab* ketiga berisi laporan kerusakan jalan oleh pengguna yang sedang *login*. Potongan kode pada tampilan utama dapat ditunjukkan pada Kode 5.3.

```
1  public class MainActivity extends AppCompatActivity
2    implementsTabLayout.OnTabSelectedListener {
3    private FirebaseAuth mAuth;
4    private CompositeDisposable compositeDisposable;
5    private FloatingActionMenu mFam;
6    private ViewPager viewPager;
7    private TabLayout tabLayout;
8    private ViewPagerAdapter viewPagerAdapter;
9    private DatabaseReference userStatusRef;
10   private int userStatus;
11   …
12   private void setupFab() {
13     mFam = findViewById(R.id.mFam);
14     userStatusRef.addValueEventListener(statusListener());
15     compositeDisposable = new CompositeDisposable();
16     FloatingActionButton fabGallery =
```

```
17        findViewById(R.id.fabGallery);
18      FloatingActionButton fabCamera =
19        findViewById(R.id.fabCamera);
20      mFam.setClosedOnTouchOutside(true);
21      Disposable fabGalleryDisposable = RxView.clicks(fabGallery)
22      .compose(new RxPermissions(MainActivity.this)
23        .ensure(Manifest.permission.READ_EXTERNAL_STORAGE))
24      .subscribe(granted->{
25        mFam.toggle(true);
26        if (granted){
27          Intent intent = new Intent(this, PostActivity.class)
28          .putExtra(Consts.TAKE_MODE_EXTRA,
29            Consts.TAKE_MODE_EXTRA_GALLERY);
30          startActivity(intent);
31        } else {
32          showPermissionReasoning();
33        }
34      });
35      Disposable fabCameraDisposable = RxView.clicks(fabCamera)
36      .subscribe(v->{
37        mFam.toggle(true);
38        Intent intent = new Intent(MainActivity.this,
39          PostActivity.class)
40        .putExtra(Consts.TAKE_MODE_EXTRA,
41          Consts.TAKE_MODE_EXTRA_CAMERA);
42        startActivity(intent);
43      });
44      compositeDisposable.addAll(fabGalleryDisposable,
45        fabCameraDisposable);
46    }
47
48    private void setupTab(){
49      tabLayout = findViewById(R.id.tab_layout);
50      viewPager = findViewById(R.id.view_pager);
51      viewPager.addOnPageChangeListener(new TabLayout
52        .TabLayoutOnPageChangeListener(tabLayout));
53      viewPagerAdapter = new ViewPagerAdapter(
54        getSupportFragmentManager(), MainActivity.this);
```

```
55      viewPager.setAdapter(viewPagerAdapter);
56      tabLayout.setupWithViewPager(viewPager);
57      tabLayout.addOnTabSelectedListener(this);
58    }
59
60    @Override
61    public boolean onOptionsItemSelected(MenuItem item) {
62      switch (item.getItemId()) {
63        case R.id.map_action_button:
64          startActivity(new Intent(this, MapsActivity.class));
65          return true;
66        case R.id.logout_submenu:
67          signOut();
68          return true;
69      }
70    }
71
72    private void showPermissionReasoning(){
73      AlertDialog.Builder alert =DisplayUtility
74        .customAlertDialog(this);
75      alert.setTitle(R.string.dialog_permission_denied)
76      .setMessage(R.string.dialog_storage_permission_reasoning)
77      .setPositiveButton(R.string.dialog_ok,null)
78      .show();
79    }
80
81    private void signOut() {
82      AlertDialog.Builder alert = DisplayUtility
83        .customAlertDialog(this);
84      alert.setTitle(R.string.dialog_confirm_signout)
85        .setPositiveButton(R.string.dialog_ok,
86      (dialog, whichButton) -> {
87        mAuth.signOut();
88        startActivity(new Intent(this, IntroActivity.class));
89        finish();
90      }).setNegativeButton(R.string.dialog_cancel, null).show();
91    }
92
```

```
93    public void startMapsActivity(LatLng latLng){
94      Bundle extras = new Bundle();
95      Intent mapsIntent = new Intent(MainActivity.this,
96        MapsActivity.class);
97      extras.putParcelable(Consts.LATLNG_EXTRA, latLng);
98      mapsIntent.putExtras(extras);
99      startActivity(mapsIntent);
100   }
101   …
102 }
```

Kode 5.3 Kode program tampilan utama

Berikut merupakan penjelasan fungsional dari potongan kode tampilan utama aplikasi pada Kode 5.3:

1. Baris 12-46 merupakan *method* untuk menyiapkan fungsi tombol *floating action menu* pada tampilan utama. *Floating action menu* berfungsi untuk membuat laporan *geotagging* kerusakan jalan baru. Ketika tombol ditekan akan menampilkan dua *floating action button*, yaitu tombol *camera* dan *gallery*.
2. Baris 21-34 berisikan kode yang dijalankan ketika *gallery floating action button* ditekan. Untuk membaca data foto dari galeri memerlukan izin akses media penyimpanan *smartphone*. Permintaan izin akses media penyimpanan menggunakan *library* RXPermissions, untuk mempersingkat kode.
3. Baris 35-43 berisikan kode yang dijalankan ketika *camera floating action button* ditekan. Aplikasi akan menavigasikan tampilan menuju *PostActivity* untuk mengambil data foto jalan rusak melalui kamera *smartphone*.
4. Baris 48-58 merupakan *method* untuk menyiapkan tampilan data laporan kerusakan jalan yang telah dilaporkan dalam bentuk *tab*. Selanjutnya konten dari masing-masing *tab* akan diatur oleh *viewpager*. Kode dari tampilan tiap tab berada pada masing-masing *fragment*.
5. Baris 60-70 merupakan implementasi *method* tombol menu pada *action bar*. Terdapat tombol peta dan tombol *logout*. Tombol peta berfungsi untuk menavigasikan tampilan menuju tampilan peta visualisasi data laporan kerusakan jalan dengan memanggil *method startMapsActivity* pada baris 93-100.
6. Baris 72-79 merupakan *method* untuk menampilkan dialog penjelasan kenapa aplikasi memerlukan izin akses media penyimpanan ketika *gallery floating action button* ditekan.
7. Baris 81-92 merupakan *method* untuk melakukan proses *logout*. Aplikasi akan menampilkan dialog konfirmasi *logout*. Jika pengguna telah menekan tombol

konfirmasi, aplikasi akan melakukan proses *logout* dengan menggunakan API Firebase *authentication*.

8. Baris 93-100 merupakan *method* untuk memanggil *MapsActivity* yang menampilkan peta visualisasi data laporan kerusakan jalan.

Untuk implementasi kode pada tiga *tab* tampilan utama memiliki *fragment* masing-masing yang diturunkan dari satu kelas abstrak *fragment*. Dari setiap *fragment* turunan, mengimplementasikan satu *method query* data sesuai dengan kebutuhan tampilan data. Berikut merupakan potongan kode kelas abstrak yang diturunkan pada setiap *fragment tab* pada Kode 5.4.

```
1  public abstract class ReportListFragment extends Fragment{
2    …
3    private ValueEventListener upvoteEventListener(){
4      return new ValueEventListener() {
5        @Override
6        public void onDataChange(DataSnapshot dataSnapshot) {
7          if (dataSnapshot.exists()){
8            userUpvotes = (HashMap<String, Boolean>)
9              dataSnapshot.getValue();
10         } else userUpvotes.clear();
11       }
12       @Override
13       public void onCancelled(DatabaseError databaseError) {}
14     };
15   }
16
17   private ValueEventListener fixedEventListener(){
18     return new ValueEventListener() {
19       @Override
20       public void onDataChange(DataSnapshot dataSnapshot) {
21         if (dataSnapshot.exists())
22           userFixedIssue = (HashMap<String, Boolean>)
23           dataSnapshot.getValue();
24         else userFixedIssue.clear();
25       }
26       @Override
27       public void onCancelled(DatabaseError databaseError) {}
28     };
29   }
30
```

```
31  private void updateVote(String reportUid, String reportKey,
32    Boolean isAdded){
33    DatabaseReference reportRef = dbRef
34      .child(Prefs.FD_REF_USERREPORTS)
35      .child(reportUid);
36    Map<String, Object> userUpvotesPath = RefPaths
37      .getVoteReportPaths(reportKey, isAdded);
38    userUpvotesRef.updateChildren(userUpvotesPath,
39      (error, ref) -> {
40      if (error == null)
41        RefPaths.runVoteTransactions(reportKey, isAdded,
42        reportRef, timelineRef);
43    });
44  }
45
46  private void menuDeleteReport(String key){
47    DisplayUtility.customAlertDialog(getActivity())
48      .setTitle(R.string.dialog_delete_report)
49      .setPositiveButton(R.string.dialog_delete,
50      ((dialog, which) -> {
51        Map<String,Object> deleteReportPaths = RefPaths
52        .getDeleteReportPaths(getUid(), key);
53        dbRef.updateChildren(deleteReportPaths);
54        String photoStorageRef = RefPaths
55        .getPhotoStorageRef(getUid())+key+".jpg";
56        FirebaseStorage.getInstance()
57          .getReference(photoStorageRef)
58          .delete();
59    }))
60    .setNegativeButton(R.string.dialog_cancel,null)
61    .show();
62  }
63
64  private void menuIssueFixed(String key){
65    Map<String, Object> fixedIssuePaths;
66    FloatingActionMenu fam = getActivity()
67      .findViewById(R.id.mFam);
68    int dialogMsgId;
```

```
69      if (userFixedIssue.get(key) == null) {
70        fixedIssuePaths = RefPaths.getFixedPaths(getUid(), key);
71        dialogMsgId = R.string.dialog_issuing_fixed_road;
72      } else {
73        fixedIssuePaths = RefPaths
74          .getCancelFixedPaths(getUid(), key);
75        dialogMsgId = R.string.dialog_cancelling_fixed_road;
76      }
77      Snackbar.make(fam, dialogMsgId,
78        Snackbar.LENGTH_SHORT).show();
79      dbRef.updateChildren(fixedIssuePaths);
80    }
81
82    private  void menuIssueAbuse(String key, Report report){
83      LayoutInflater li = LayoutInflater.from(getActivity());
84      View abuseDialogLayout = li.inflate(
85        R.layout.dialog_abuse_reasoning, null);
86      final EditText reasoningET = abuseDialogLayout
87        .findViewById(R.id.reasoning_input);
88      DisplayUtility.customAlertDialog(getActivity())
89      .setView(abuseDialogLayout)
90      .setNegativeButton(R.string.dialog_cancel,null)
91      .setPositiveButton(R.string.dialog_ok, (dialog, which) -> {
92        String reasoning = reasoningET.getText().toString();
93        Map<String, Object> abuseIssuePaths;
94        abuseIssuePaths = RefPaths.getAbuseIssuePaths(
95          getUid(), mName, report, key, reasoning);
96        dbRef.updateChildren(abuseIssuePaths);
97      }).show();
98    }
99
100   public abstract Query getQuery(DatabaseReference qRef);
101   …
102 }
```

Kode 5.4 Kode kelas abstrak *fragment* tampilan utama

Berikut merupakan penjelasan secara detail dari Kode 5.4 kelas abstrak *fragment* tampilan utama:

1. Baris 3-15 merupakan *method* yang mengembalikan *EventListener* untuk menampilkan perubahan data *vote* dari laporan *geotagging* kerusakan jalan.
2. Baris 17-29 merupakan *method* yang mengembalikan *EventListener* untuk menampilkan perubahan data ketika laporan *geotagging* kerusakan jalan telah dikonfirmasi oleh admin bahwa jalan yang dilaporkan telah diperbaiki.
3. Baris 31-44 merupakan *method* yang berfungsi untuk melakukan transaksi data menuju Firebase RDB ketika pengguna menekan tombol *vote* pada laporan *geotagging* kerusakan jalan.
4. Baris 46-62 merupakan *method* yang dijalankan ketika pengguna aplikasi *smartphone* menghapus laporan *geotagging* kerusakan jalan. Ketika tombol hapus laporan ditekan, akan ditampilkan dialog konfirmasi. Jika pengguna memberikan konfirmasi penghapusan, maka akan dilakukan transaksi data menuju server Firebase RDB untuk menghapus data laporan *geotagging* kerusakan jalan pada server.
5. Baris 64-80 merupakan *method* yang dijalankan ketika pengguna menekan tombol menu isukan jalan diperbaiki. Dalam *method* ini dilakukan transaksi data menuju Firebase RDB untuk melakukan perubahan data isu jalan telah diperbaiki. Jika pengguna telah mengisukan jalan diperbaiki, tombol menu akan berubah menjadi batalkan isu jalan diperbaiki. Setelah proses transaksi data berhasil, akan ditampilkan pesan dalam bentuk *snackbar*.
6. Baris 82-98 merupakan *method* untuk mengisukan penyalahgunaan laporan *geotagging* kerusakan jalan. Setelah tombol isukan laporan ditekan, akan ditampilkan dialog konfirmasi dengan isian alasan penyalahgunaan laporan yang diisukan. Jika pengguna mengkonfirmasi, akan dilakukan transaksi data menuju Firebase RDB untuk menambahkan data isu penyalahgunaan.
7. Baris 100 merupakan *method* abstrak yang diimplementasikan oleh *fragment* turunan. *Method* ini harus mengembalikan nilai dengan tipe data *Query* dan menerima parameter *node reference* dari Firebase RDB. *Query* yang diimplementasikan disesuaikan dengan kebutuhan tampilan data pada masing-masing *tab* laporan *geotagging* laporan kerusakan jalan.

5.4.2.1 Kode Program Data Lini Masa

```
1  public class MainTimelineFragment extends ReportListFragment {
2      public MainTimelineFragment(){}
3      @Override
4      public Query getQuery(DatabaseReference qRef){
5        return qRef.child(Prefs.FD_REF_REPORTS);
6      }
7  }
```

Kode 5.5 Kode program kelas turunan untuk *fragment* lini masa

Kode 5.5 merupakan kelas turunan dari kelas abstrak *fragment* untuk tampilan utama. Pada *fragment* ini diimplementasikan *method getQuery* yang mengembalikan semua data pada *node Reports* untuk menampilkan semua data laporan kerusakan jalan berdasarkan urutan waktu, data terbaru pada urutan teratas.

5.4.2.2 Kode Program Data Jalan Telah Diperbaiki

```
1  public class MainFixedFragment extends ReportListFragment {
2      public MainFixedFragment(){}
3      @Override
4      public Query getQuery(DatabaseReference qRef){
5          return qRef.child(Prefs.FD_REF_REPORTS)
6              .orderByChild("fixed").equalTo(true);
7      }
8  }
9
```

Kode 5.6 Kode program kelas turunan untuk *fragment* jalan telah diperbaiki

Kode 5.6 adalah *fragment* untuk tampilan jalan telah diperbaiki yang diturunkan dari *fragment* tampilan utama. *Fragment* ini diperlukan agar pengguna dapat dengan mudah melihat data laporan *geotagging* kerusakan jalan yang telah diperbaiki. *Method* yang diimplementasikan pada kelas ini mengembalikan *query* pada *node Reports* yang memiliki nilai *true* pada properti *fixed* dari laporan *geotagging* kerusakan jalan.

5.4.2.3 Kode Program Data Laporan Pengguna

```
1  public class MainMyReportFragment extends ReportListFragment {
2      public MainMyReportFragment(){}
3      @Override
4      public Query getQuery(DatabaseReference qRef){
5          return qRef.child(Prefs.FD_REF_USERREPORTS)
6              .child(getUid());
7      }
8  }
9
```

Kode 5.7 Kode program kelas turunan untuk *fragment* laporan pengguna

Kode 5.7 menunjukkan potongan kode *fragment* yang digunakan untuk menampilkan data laporan pengguna yang sedang *login* pada aplikasi *geotagging* laporan kerusakan jalan. Tampilan ini diperlukan agar pengguna dapat dengan mudah melihat laporannya sendiri untuk melakukan penghapusan. *Method* yang

diimplementasikan mengembalikan *query* untuk data pada *node UserReports* yang disusun berdasarkan data identifikasi pengguna.

5.4.3 Kode Program Buat Laporan Baru

Pada aplikasi *geotagging* laporan kerusakan jalan pengguna dapat membuat laporan baru melalui dua cara. Yang pertama dengan mengambil data foto jalan rusak dari galeri, yang kedua mengambil data foto jalan rusak dari kamera *smartphone* secara langsung. Berikut merupakan potongan kode untuk membuat laporan *geotagging* baru pada Kode 5.8.

```
1  public class PostActivity extends AppCompatActivity
2  implements View.OnClickListener{
3    …
4    private void cameraAction(){
5      setThumbnailView(Uri.fromFile(photoFile));
6      Disposable camDisposable = Observable
7      .just(Consts.STRING_NOT_AVAILABLE)
8      .subscribeOn(Schedulers.io())
9      .observeOn(Schedulers.io())
10     .doOnNext(x->{
11       photoUri = Uri.fromFile(photoFile);
12       photoTaken = BitmapFactory.decodeFile(photoPath);
13     })
14     .compose(aggregatorObservableTransformer())
15     .subscribe();
16     compositeDisposable.add(camDisposable);
17   }
18
19   private void galleryAction(Uri uri){
20     setThumbnailView(uri);
21     Disposable galDisposable = Observable
22     .just(Consts.STRING_NOT_AVAILABLE)
23     .subscribeOn(Schedulers.io())
24     .observeOn(Schedulers.io())
25     .doOnNext(x -> {
26       photoUri = uri;
27       try {
28         photoTaken = MediaStore.Images.Media
29         .getBitmap(PostActivity.this.getContentResolver(),
30         photoUri);
```

```
31        } catch (IOException e) {
32          e.printStackTrace();
33        }
34      })
35      .compose(aggregatorObservableTransformer())
36      .subscribe();
37      compositeDisposable.add(galDisposable);
38    }
39    …
40    private ObservableTransformer<String,String>
41    writePhotoObservableTransformer(){
42      return observable -> observable
43      .doOnNext(x -> {
44        Bitmap compressedBitmap = PhotoUtility
45        .rotateBitmap(PostActivity.this, photoUri, photoTaken);
46        ByteArrayOutputStream bytes = new
47        ByteArrayOutputStream();
48        compressedBitmap.compress(Bitmap.CompressFormat.JPEG,
49          Consts.PHOTO_QUALITY, bytes);
50        File resizedFile = new File(photoPath);
51        try {
52          FileOutputStream fos = new
53            FileOutputStream(resizedFile);
54          fos.write(bytes.toByteArray());
55          fos.close();
56        } catch (IOException e) {
57          e.printStackTrace();
58        }
59        thumbnailView.setOnClickListener(this);
60      });
61    }
62
63    private ObservableTransformer<String,String>
64    exifObservableTransformer(){
65      return observable -> observable
66      .map(x -> {
67        mLatLng = PhotoUtility.getExifLocation(photoPath);
68        if (mLatLng != null) setStreetName(mLatLng);
```

```
69        return streetName;
70      })
71      .observeOn(AndroidSchedulers.mainThread())
72      .doOnNext(street -> {
73        if (!street.equals(Consts.STRING_NOT_AVAILABLE)) {
74          setLocationFound(street,true);
75          customLocationButton.setVisibility(View.VISIBLE);
76          mFab.show();
77        } else {
78          tvStreetName.setText(R.string.get_gps_location);
79        customLocationButton.setVisibility(View.VISIBLE);
80      });
81    }
82
83    private Observable<String> getCurrentLocation(){
84      locationRequest = LocationRequest.create()
85        .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
86        .setNumUpdates(1).setInterval(3000);
87      rxLocation = new RxLocation(PostActivity.this);
88      rxLocation.setDefaultTimeout(10, TimeUnit.SECONDS);
89      return rxLocation.settings()
90        .checkAndHandleResolution(locationRequest)
91        .flatMapObservable(isActivated -> {
92          if (isActivated)
93            return onLocationSettingsActivatedObservable();
94          return onLocationNotFoundObservable();
95      });
96    }
97    …
98    private void setStreetName(LatLng latLng){
99      List<Address> addresses = null;
100     try {
101       addresses = new Geocoder(PostActivity.this)
102       .getFromLocation(latLng.latitude, latLng.longitude,1);
103     } catch (IOException e) { e.printStackTrace(); }
104     streetName = addresses.get(0).getThoroughfare();
105   }
106   …
```

```
107   private void postReport(){
108     AppCompatEditText descEditText = findViewById(
109       R.id.text_input_description);
110     String description = descEditText.getText().toString();
111     String uid = mAuth.getUid();
112     HashMap<String, Double> coord = new HashMap<>();
113     coord.put(Consts.KEY_LATITUDE, mLatLng.latitude);
114     coord.put(Consts.KEY_LONGITUDE, mLatLng.longitude);
115     sp = PreferenceManager
116       .getDefaultSharedPreferences(PostActivity.this);
117     userName = sp.getString(Prefs.PREF_KEY_USER_NAME,
118       Consts.STRING_NOT_AVAILABLE);
119     DatabaseReference dbRef = FbPersistence.getDatabase()
120       .getReference();
121     String key = dbRef.child("Reports").push().getKey();
122     compositeDisposable.add(uploadPhoto(uid, key)
123       .flatMapCompletable(snap -> {
124         Report report = new Report(uid, userName, streetName,
125           description, snap.getDownloadUrl()
126           .toString(), coord);
127         return saveReportDataCompletable(dbRef, key, report);
128       }).subscribe());
129   }
130   …
131 }
```

Kode 5.8 Kode program untuk membuat laporan baru

Dalam pembuatan laporan *geotagging* baru dibutuhkan data foto jalan rusak, data lokasi kerusakan jalan, dan deskripsi yang bersifat opsional. Pemrosesan data pada pembuatan laporan geotagging baru dilakukan menggunakan *library* RxJava untuk memudahkan implementasi kode. Dengan menggunakan RxJava, pemrosesan data dapat dimodelkan sebagai aliran proses. Berikut ini adalah penjelasan lebih detail mengenai potongan kode pada Kode 5.8 untuk membuat laporan *geotagging* kerusakan jalan:

1. Baris 4-17 merupakan kode untuk melakukan pemrosesan gambar yang diambil dari kamera *smartphone*. Gambar ditampilkan pada pratinjau menggunakan *library* Glide. Kemudian data gambar dialirkan dalam pemrosesan menggunakan RxJava.

2. Baris 19-38 berfungsi untuk melakukan pemrosesan gambar yang diambil dari galeri. Gambar yang dipilih akan ditampilkan dalam pratinjau gambar, kemudian data gambar dialirkan dalam pemrosesan menggunakan RxJava.
3. Data gambar yang didapat dari kamera maupun galeri dialirkan menuju pemrosesan pada baris kode 40-61. Data gambar yang diambil akan diperkecil ukurannya agar efisien dalam pemrosesan dan pengunggahan data. Selain itu, gambar juga diputar sesuai dengan orientasi *smartphone* ketika mengambil gambar. Data gambar disimpan menggunakan *fileprovider* untuk melakukan pemrosesan selanjutnya
4. Baris 63-81 berfungsi untuk membaca data lokasi pada EXIF gambar. Jika gambar memiliki data EXIF lokasi, maka data lokasi tersebut akan diterjemahkan menjadi nama jalan pada baris kode 98-105. Jika tidak ditemukan, pemrosesan akan dialirkan pada baris kode 83-96.
5. Baris 83-96 berfungsi untuk mengambil data lokasi pengguna saat ini jika tidak ditemukan data lokasi pada EXIF gambar. Untuk mendapatkan data lokasi pengguna memerlukan sensor GPS pada *smartphone*. Jika sensor GPS dalam keadaan tidak aktif, maka akan muncul dialog untuk mengaktifkan sensor. Jika sensor GPS telah aktif, dilakukan pengambilan data lokasi yang akan diterjemahkan menjadi nama jalan pada baris kode 98-105
6. Baris 98-105 berfungsi untuk menerjemahkan data lokasi berupa *latitude* dan *longitude* menjadi nama jalan menggunakan API Google *geocoding*. Jika nama jalan ditemukan, tombol unggah laporan akan dimunculkan.
7. Baris 107-129 berfungsi untuk mengunggah laporan *geotagging* kerusakan jalan. Data gambar diunggah ke layanan Firebase Storage, setelah gambar terunggah dilakukan transaksi penulisan detail data laporan *geotagging* kerusakan jalan pada Firebase RDB.

5.4.4 Kode Program Peta Visualisasi Data

Kode 5.9 merupakan potongan kode program aplikasi *geotagging* laporan kerusakan jalan untuk menampilkan peta visualisasi data berdasarkan lokasi laporan kerusakan jalan.

```
1  public class MapsListFragment extends Fragment
2  implements View.OnClickListener, OnMapReadyCallback{
3    …
4    @Override
5    public void onCreate(@Nullable Bundle savedInstanceState) {
6      super.onCreate(savedInstanceState);
7      getActivity().setTitle(R.string.maps_cluster_mode);
8      mLatLng = Consts.MALANG_LATLNG;
9      if (getArguments() != null) {
10       mLatLng = getArguments()
```

```
        .getParcelable(Consts.LATLNG_EXTRA);
      initialLatLng = true;
    }
    DatabaseReference dbRef = FbPersistence.getDatabase()
      .getReference();
    dbMapsRef = dbRef.child(Prefs.FD_REF_REPORTS);
    dbMapsRef.keepSynced(true);
    compositeDisposable = new CompositeDisposable();
  }
  …
  @Override
  public void onMapReady(GoogleMap mMap) {
    googleMap = mMap;
    try {
      googleMap.setMapStyle(MapStyleOptions
        .loadRawResourceStyle(getActivity()
        .getBaseContext(), R.raw.map_style_dark));
    } catch (Resources.NotFoundException e) {
      Snackbar.make(fam, R.string.style_not_found,
        Snackbar.LENGTH_SHORT).show();
    }
    googleMap.getUiSettings().setMapToolbarEnabled(false);
    googleMap.getUiSettings().setMyLocationButtonEnabled(false);
    googleMap.getUiSettings().setCompassEnabled(true);
    googleMap.setMyLocationEnabled(true);
    googleMap.setMaxZoomPreference(20f);
    googleMap.setMinZoomPreference(11f);
    googleMap.setLatLngBoundsForCameraTarget(
        Consts.MALANG_BOUNDS);
    if (!initialLatLng) {
      setMapsCamera(mLatLng, 13.5f, false);
      compositeDisposable.add(setMapsCameraCompletable(
        13.5f, false).subscribe());
    } else setMapsCamera(mLatLng, 16, false);
    compositeDisposable.add(
      mapEventListenerFlowable().subscribe());
  }
  …
```

```
49  private void setupClustering(){
50    if (mClusterManager==null) mClusterManager =
51      new ClusterManager<>(getActivity()
52        .getBaseContext(), googleMap);
53    populateCluster();
54
55    mClusterManager.getMarkerCollection()
56      .setOnInfoWindowAdapter(
57        new ClusterInfoWindowAdapter(mapItems,
58        getActivity()));
59    googleMap.setInfoWindowAdapter(mClusterManager
60     .getMarkerManager());
61    googleMap.setOnCameraIdleListener(mClusterManager);
62    googleMap.setOnMarkerClickListener(mClusterManager);
63    mClusterManager.setOnClusterClickListener(cluster -> {
64      float zoom = googleMap.getCameraPosition().zoom+1.5f;
65      setMapsCamera(cluster.getPosition(), zoom, true);
66      return true;
67    });
68    mClusterManager.setAnimation(true);
69    mClusterManager.cluster();
70  }
71
72  private void setupHeatMap(){
73    populateHeatMap();
74    if (heatMapProvider==null)
75      heatMapProvider = new HeatmapTileProvider.Builder()
76        .radius(23).weightedData(heatMapList).build();
77    if (!DisplayUtility.isDay()) heatMapProvider
78      .setGradient(getCustomGradient());
79    heatMapProvider.setOpacity(0.55d);
80    heatMapOverlay = googleMap.addTileOverlay(new
81      TileOverlayOptions().tileProvider(heatMapProvider));
82  }
83  …
84  private Single<Boolean> myLocationObservable(){
85    locationRequest = LocationRequest.create()
86    .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY)
```

```
87      .setNumUpdates(1)
88      .setInterval(3000);
89      rxLocation = new RxLocation(getActivity()
90        .getBaseContext());
91      rxLocation.setDefaultTimeout(10, TimeUnit.SECONDS);
92      return rxLocation.settings()
93      .checkAndHandleResolution(locationRequest);
94    }
95    …
96    private void setMapsCamera(LatLng latLng,
97    float zoom,  Boolean animate){
98      CameraUpdate cameraUpdate;
99      if (zoom > 20) zoom = 20f;
100     if (zoom >= 12) cameraUpdate = CameraUpdateFactory
101       .newCameraPosition(
102         new CameraPosition.Builder()
103         .target(latLng).zoom(zoom).build());
104     else cameraUpdate = CameraUpdateFactory
105       .newLatLng(latLng);
106     if (animate) googleMap.animateCamera(cameraUpdate,
107       800, null);
108     else googleMap.moveCamera(cameraUpdate);
109   }
110   …
111 }
```

Kode 5.9 Kode program untuk peta visualisasi data kerusakan jalan

Berikut penjelasan lebih detail mengenai potongan kode program untuk menampilkan peta visualisasi data kerusakan jalan pada Kode 5.9 :

1. Baris 4-19 adalah *method* untuk membaca data parameter lokasi. Jika parameter lokasi tidak ditemukan, peta akan menampilkan koordinat lokasi Kota Malang.
2. Baris 21-47 adalah *method* untuk menyiapkan tampilan peta. Pada *method* ini diatur fitur peta apa saja yang akan ditampilkan
3. Baris 49-70 adalah *method* untuk menyiapkan data visualisasi dalam bentuk *clustered marker.*
4. Baris 72-82 adalah *method* untuk menyiapkan data visualisasi dalam bentuk *heatmap*.

5. Baris 84-94 adalah *method* untuk mengambil data lokasi pengguna saat ini untuk ditampilkan pada peta.
6. Baris 96-109 adalah *method* untuk memindahkan kamera peta sesuai dengan parameter yang dimasukkan.

5.5 Implementasi Antarmuka

Implementasi antarmuka aplikasi *geotagging* laporan kerusakan jalan pada perangkat Android dijelaskan pada tahap ini. Antarmuka diimplementasikan menggunakan tampilan *material design*.

5.5.1 Antarmuka Halaman Pendaftaran dan *Login*

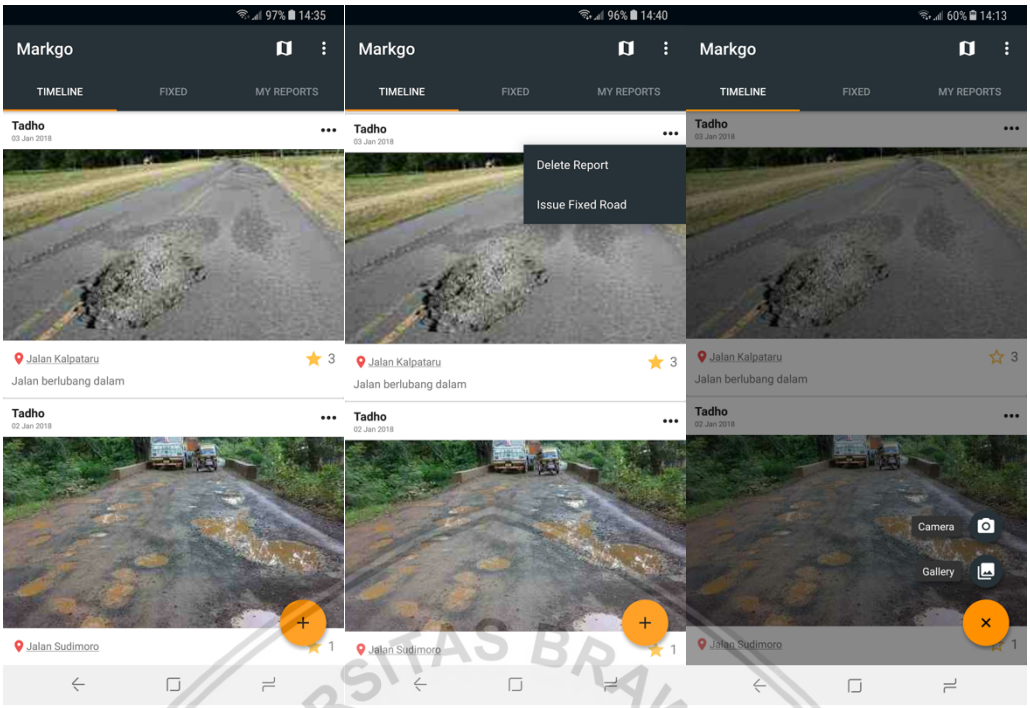
Pengguna dapat melakukan proses pendaftaran atau *login* untuk dapat masuk dan menggunakan aplikasi. Pada halaman pendaftaran dan *login*, ditampilkan dalam tampilan *slide*. Terdapat tiga tampilan *slide* yang menampilkan logo dan deskripsi singkat aplikasi, permohonan izin penggunaan sensor GPS, dan isian identitas pengguna untuk melakukan pendaftaran atau *login*. Gambar 5.1 menunjukkan tampilan antarmuka halaman pendaftaran dan *login*.

Gambar 5.1 Antarmuka halaman pendaftaran dan *login*

Pada *slide* pertama, tampilan antarmuka menggunakan skema warna utama aplikasi, yaitu corak warna abu-abu kehitaman yang identik dengan warna jalan. Warna pada *slide* kedua dan ketiga berbeda untuk menambah aksen warna aplikasi yang menggunakan pilihan corak warna *material design*.

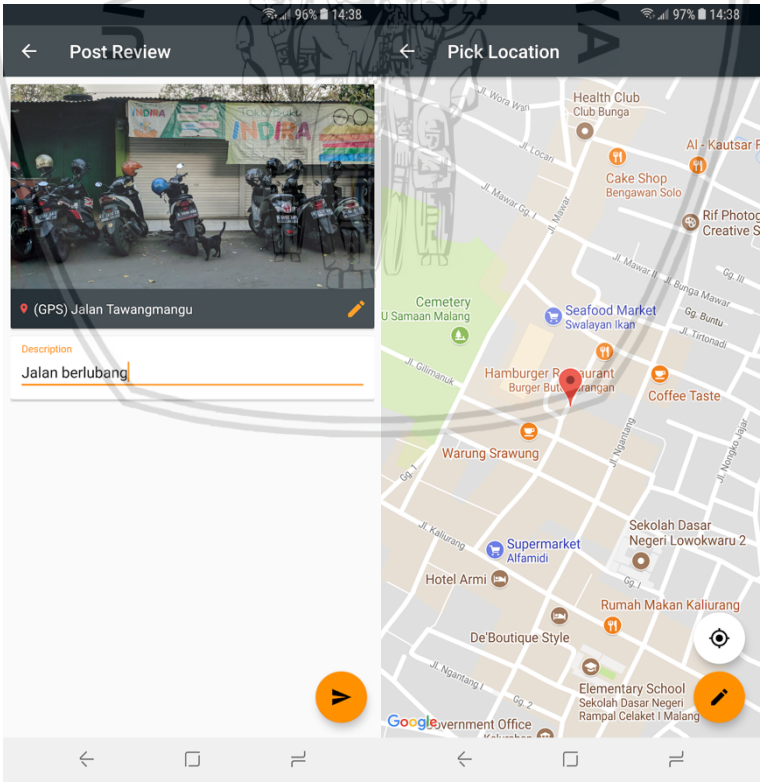
5.5.2 Antarmuka Halaman Utama

Pada halaman utama berisi data laporan *geotagging* kerusakan jalan. Di halaman ini terdapat tombol untuk berinteraksi dengan laporan *geotagging* kerusakan jalan seperti memberikan *vote*, mengisukan jalan diperbaiki, mengisukan penyalahgunaan, menghapus laporan sendiri, dan juga tombol untuk membuat laporan *geotagging* kerusakan jalan baru. Gambar 5.2 menunjukkan implementasi antarmuka halaman pada aplikasi.



Gambar 5.2 Antarmuka halaman utama

5.5.3 Antarmuka Halaman Pratinjau Laporan Baru

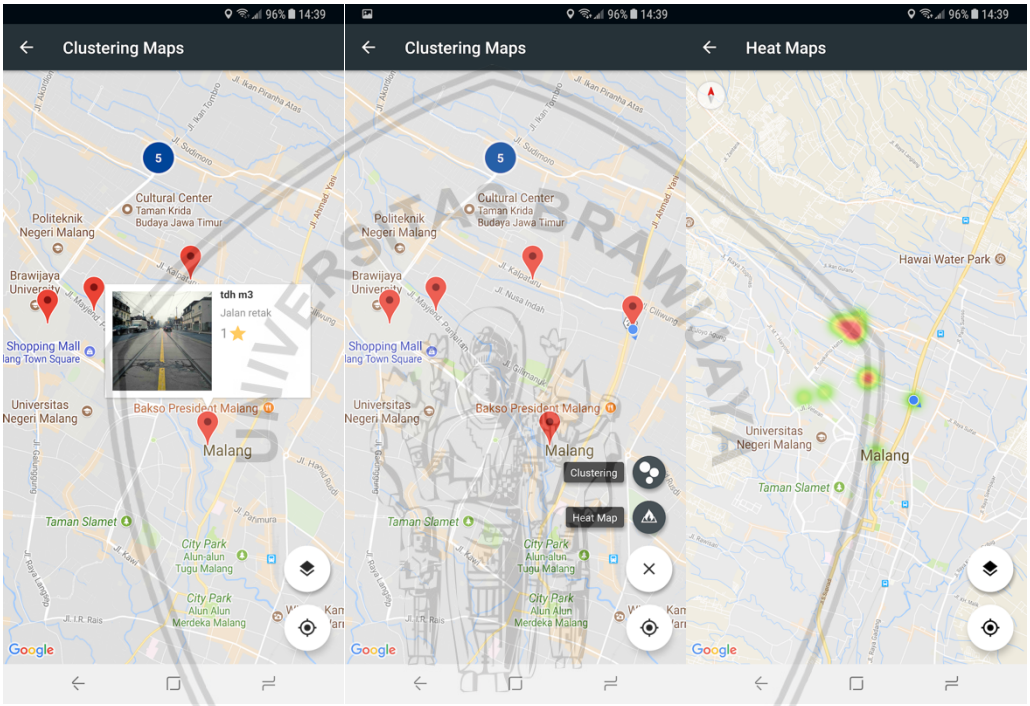


Gambar 5.3 Antarmuka halaman pratinjau laporan baru

Pada Gambar 5.3 pengguna dapat meninjau laporan yang akan diunggah. Terdapat tampilan pratinjau foto, lokasi yang berupa nama jalan, dan kotak isian

deskripsi laporan. Selain itu ada tombol untuk mengganti lokasi secara manual di sebelah kanan nama jalan yang mengarahkan tampilan menuju tampilan *pick location*. Pada tampilan *pick location* pengguna dapat memilih lokasi dengan cara menekan suatu lokasi pada tampilan peta, setelah itu dapat menekan tombol *floating action button* yang memiliki ikon pensil untuk menerapkan lokasi yang dipilih. Deskripsi laporan *geotagging* kerusakan jalan bersifat opsional. Laporan hanya membutuhkan data foto dan nama jalan untuk dapat diunggah. Jika lokasi tidak ditemukan dan belum diisikan oleh pengguna maka tombol *floating action button* untuk mengunggah laporan tidak ditampilkan.

5.5.4 Antarmuka Halaman Peta Visualisasi Data



Gambar 5.4 Antarmuka halaman peta visualisasi data

Gambar 5.4 menunjukkan implementasi antarmuka halaman peta visualisasi data laporan *geotagging* kerusakan jalan. Pada halaman ini lokasi jalan yang dilaporkan rusak, divisualisasikan pada peta. Ada dua mode visualisasi data, yaitu *clustered-marker* dan *heatmap*. Mode *clustered-marker* menampilkan penanda untuk merepresentasikan lokasi laporan *geotagging* kerusakan jalan. Jika lokasi saling berdekatan, penanda akan digabungkan dan ditampilkan berapa jumlah laporan kerusakan jalan pada daerah yang direpresentasikan. Masing-masing penanda dapat ditekan untuk melihat nama pelapor, foto kerusakan jalan, deskripsi laporan, dan jumlah *vote* pada laporan. Mode *heatmap* merepresentasikan laporan kerusakan jalan dalam bentuk warna gradien. Tingkat gelap terang warna gradien berdasarkan jumlah laporan dan *vote* pada laporan. Jika semakin banyak laporan pada suatu lokasi atau suatu laporan memilki jumlah *vote* yang banyak, warna *heatmap* semakin gelap.

BAB 6 PENGUJIAN DAN ANALISIS

Pengujian dan analisis aplikasi perangkat bergerak *geotagging* laporan kerusakan jalan dibahas pada bab ini. Tujuan dari dilakukannya pengujian adalah untuk dapat menemukan ketidaksesuaian aplikasi perangkat bergerak *geotagging* laporan kerusakan jalan yang telah dirancang dan dibangun baik dari sisi fungsional maupun non-fungsional. Dari hasil pengujian akan dilakukan analisis untuk menemukan apakah target dari penelitian ini tercapai.

6.1 Pengujian

Pengujian dilakukan terhadap aplikasi pada perangkat bergerak *geotagging* laporan kerusakan jalan. Akan dilakukan dua proses pengujian terhadap aplikasi, yaitu pengujian validasi dan pengujian *usability*. Pengujian validasi menggunakan teknik *blackbox testing*, sedangkan untuk pengujian *usability* menggunakan kuesioner yang diisikan oleh pengguna aplikasi.

6.1.1 Pengujian validasi

Tujuan dari pengujian validasi adalah untuk mengetahui apakah aplikasi berfungsi sebagaimana yang telah dirancang sebelumnya pada analisis kebutuhan. Pada pengujian validasi ini, digunakan teknik *blackbox testing* untuk menemukan ketidaksesuaian fungsional aplikasi dengan rancangannya. Pengujian validasi terdiri dari beberapa kasus uji yang masing-masing memiliki kode identifikasi kasus uji, nama kasus uji, kode identifikasi *usecase* yang direpresentasikan pada kasus tersebut, tujuan pengujian masing-masing kasus, prosedur pengujian, dan hasil yang diharapkan pada aplikasi.

Tabel 6.1 Kasus uji membuat laporan baru dari kamera

ID Kasus Uji	**TC-M01**
Nama Kasus Uji	Membuat laporan baru dari kamera
ID *Use Case*	**SRS-M01**
Tujuan Pengujian	Membuktikan jika fitur aplikasi *mobile* untuk membuat laporan kerusakan jalan baru menggunakan sensor kamera berfungsi
Prosedur Pengujian	1. Menekan *floating action menu* pada halaman utama 2. Menekan *camera floating action button* 3. Ambil gambar jalan rusak 4. Mengisi data lokasi kerusakan jalan 5. Mengisi deskripsi 6. Menekan *floating action button* unggah laporan 7. Cek ketersediaan data pada server Firebase

Tabel 6.1 Kasus uji membuat laporan baru (lanjutan)

Hasil yang Diharapkan	Serangkaian proses untuk membuat laporan baru berfungsi dengan normal, seperti fungsi mengambil gambar melalui sensor kamera *smartphone*, data lokasi, mengunggah data, sehingga data tersimpan dengan benar pada Firebase RDB.

Tabel 6.2 Kasus uji membuat laporan baru dari galeri

ID Kasus Uji	**TC-M02**
Nama Kasus Uji	Membuat laporan baru dari galeri
ID *Use Case*	**SRS-M01**
Tujuan Pengujian	Membuktikan jika fitur aplikasi *mobile* untuk membuat laporan kerusakan jalan baru dari gambar yang ada di galeri
Prosedur Pengujian	1. Menekan *floating action menu* pada halaman utama 2. Menekan *gallery floating action button* 3. Pilih gambar jalan rusak 4. Mengisi data lokasi kerusakan jalan 5. Mengisi deskripsi 6. Menekan *floating action button* unggah laporan 7. Cek ketersediaan data pada server Firebase
Hasil yang Diharapkan	Serangkaian proses untuk membuat laporan baru berfungsi dengan normal, seperti fungsi memilih gambar dari galeri *smartphone*, data lokasi, mengunggah data, sehingga data tersimpan dengan benar pada server Firebase RDB.

Tabel 6.3 Kasus uji melihat data laporan *geotagging* kerusakan jalan

ID Kasus Uji	**TC-M03**
Nama Kasus Uji	Melihat data laporan
ID *Use Case*	**SRS-M02**
Tujuan Pengujian	Membuktikan jika data laporan *geotagging* kerusakan jalan yang telah dilaporkan pada aplikasi dapat dilihat oleh pengguna

Tabel 6.3 Kasus uji melihat data laporan *geotagging* kerusakan jalan (lanjutan)

Prosedur Pengujian	1. *Login* pada aplikasi 2. Buat laporan *geotagging* kerusakan jalan baru 3. Cek tampilan setiap *tab* pada halaman utama aplikasi
Hasil yang Diharapkan	Data laporan *geotagging* kerusakan jalan dapat ditampilkan sesuai dengan rancangan.

Tabel 6.4 Kasus uji melihat peta visualisasi data

ID Kasus Uji	**TC-M04**
Nama Kasus Uji	Melihat peta visualisasi data
ID *Use Case*	**SRS-M03**
Tujuan Pengujian	Membuktikan jika fitur aplikasi *mobile* untuk melihat peta visualisasi laporan kerusakan jalan berfungsi
Prosedur Pengujian	1. Tekan tombol dengan ikon peta pada *action bar* 2. Tekan tombol konfirmasi untuk menyalakan sensor GPS 3. Tekan salah satu *marker* laporan kerusakan jalan 4. Tekan *layer floating action button* untuk mengganti mode visualisasi peta
Hasil yang Diharapkan	Visualisasi *clustered marker* dapat menampilkan laporan *geotagging* kerusakan jalan sesuai dengan lokasi yang dilaporkan. Jika ada beberapa laporan *geotagging* kerusakan jalan dengan lokasi yang berdekatan, akan digabungkan menjadi satu *clustered marker*. Untuk visualisasi *heatmap*, tingkat intensitas gradien warna sebanding dengan jumlah laporan dan jumlah *vote* pada lokasi laporan *geotagging* kerusakan jalan.

Tabel 6.5 Kasus uji hapus laporan

ID Kasus Uji	**TC-M05**
Nama Kasus Uji	Menghapus data laporan
ID *Use Case*	**SRS-M04**
Tujuan Pengujian	Membuktikan jika fitur aplikasi *mobile* untuk menghapus laporan berfungsi

Tabel 6.5 Kasus uji hapus laporan (lanjutan)

Prosedur Pengujian	1. Tekan tombol menu pada laporan *geotagging* kerusakan jalan oleh pengguna yang sedang *login* sekarang 2. Tekan menu "*Delete Report*" 3. Konfirmasi dialog penghapusan data laporan *geotagging* kerusakan jalan
Hasil yang Diharapkan	Data laporan *geotagging* kerusakan jalan dapat terhapus dengan benar, termasuk data redundansi untuk *node* tambahan.

Tabel 6.6 Kasus uji mengisukan jalan yang telah diperbaiki

ID Kasus Uji	**TC-M06**
Nama Kasus Uji	Mengisukan jalan diperbaiki
ID *Use Case*	**SRS-M05**
Tujuan Pengujian	Membuktikan jika fitur aplikasi *mobile* untuk mengisukan/membatalkan isu jalan diperbaiki dapat berfungsi
Prosedur Pengujian	1. Tekan tombol menu pada laporan *geotagging* kerusakan jalan 2. Tekan menu "*Issue Fixed Road*" 3. Tekan menu "*Cancel Fixed Road*"
Hasil yang Diharapkan	Aplikasi menampilkan pesan konfirmasi bahwa laporan *geotagging* kerusakan jalan telah berhasil/batal diisukan untuk diperbaiki dan data tersimpan pada Firebase RDB.

Tabel 6.7 Kasus uji mengisukan penyalahgunaan

ID Kasus Uji	**TC-M07**
Nama Kasus Uji	Mengisukan penyalahgunaan laporan
ID *Use Case*	**SRS-M06**
Tujuan Pengujian	Membuktikan jika fitur aplikasi *mobile* untuk mengisukan/membatalkan isu penyalahgunaan laporan *geotagging* kerusakan jalan.

Tabel 6.7 Kasus uji mengisukan penyalahgunaan (lanjutan)

Prosedur Pengujian	1. Tekan tombol menu pada laporan *geotagging* kerusakan jalan 2. Tekan menu "*Issue Abusive Post*" 3. Isikan deskripsi penyalahgunaan 4. Tekan tombol "*OK*"
Hasil yang Diharapkan	Aplikasi dapat menampilkan dialog konfirmasi dan kotak isian alasan penyalahgunaan, kemudian data isu penyalahgunaan dikirim dan tersimpan pada Firebase RDB untuk ditampilkan pada aplikasi *web* admin

Tabel 6.8 Kasus uji *vote* laporan

ID Kasus Uji	**TC-M08**
Nama Kasus Uji	*Vote* laporan
ID *Use Case*	**SRS-M07**
Tujuan Pengujian	Membuktikan jika fitur untuk memberikan *vote* pada laporan *geotagging* kerusakan jalan dapat berfungsi
Prosedur Pengujian	1. Tekan tombol dengan ikon bintang dibagian kanan bawah foto laporan *geotagging* kerusakan jalan 2. Tekan tombol lagi untuk membatalkan *vote*
Hasil yang Diharapkan	Aplikasi menampilkan ikon bintang yang terisi warna kuning untuk laporan yang telah di*vote* oleh pengguna. Untuk aplikasi yang belum/batal di*vote*, aplikasi menampilkan ikon bintang yang tidak terisi warna kuning.

Berikutnya ditentukan hasil pengujian berdasarkan kasus uji yang dibahas pada Tabel 6.1 sampai dengan Tabel 6.8. Pada tabel kasus uji ditunjukkan tujuan pengujian, prosedur pengujian, dan hasil yang diharapkan pada aplikasi yang telah dirancang dan dibangun. Kemudian aplikasi perangkat bergerak diuji sesuai dengan prosedur pengujian pada tabel kasus uji, sehingga didapatkan hasil pengujian. Dari hasil pengujian yang didapat, akan dibandingkan dengan hasil yang diharapkan. Jika hasil yang diharapkan sesuai dengan hasil yang didapat pada saat pengujian validasi pada aplikasi perangkat bergerak, maka status hasil pengujian pada kasus uji tersebut dapat dinyatakan valid. Untuk tabel data hasil pengujian validasi ditunjukkan pada Tabel 6.9.

Tabel 6.9 Hasil pengujian validasi

ID	Hasil Yang Diharapkan	Hasil Yang Diperoleh	Status
TC-M01	Serangkaian proses untuk membuat laporan baru berfungsi dengan normal, seperti fungsi mengambil gambar melalui sensor kamera *smartphone*, data lokasi, mengunggah data, hingga data tersimpan dengan benar pada Firebase RDB	Sistem dapat menjalankan serangkaian proses untuk membuat laporan baru melalui sensor kamera dan data dapat tersimpan pada Firebase RDB	Valid
TC-M02	Serangkaian proses untuk membuat laporan baru berfungsi dengan normal, seperti fungsi memilih gambar dari galeri *smartphone*, data lokasi, mengunggah data, sehingga data tersimpan dengan benar pada server Firebase RDB	Sistem dapat menjalankan serangkaian proses untuk membuat laporan baru dari gambar yang dipilih pada galeri dan data dapat tersimpan pada Firebase RDB	Valid
TC-M03	Data laporan *geotagging* kerusakan jalan dapat ditampilkan sesuai dengan rancangan	Sistem dapat menampilkan data laporan *geotagging* kerusakan jalan sesuai dengan rancangan	Valid
TC-M04	Visualisasi data berupa *clustered marker* dan *heatmap* ditampilkan oleh sistem sesuai dengan rancangan tampilan. Tombol untuk mengganti mode visualisasi *clustered marker* dan *heatmap* dapat dijalankan	Sistem dapat menampilkan visualisasi data sesuai dengan rancangan dan tombol mode visualisasi berfungsi dengan benar	Valid
TC-M05	Data laporan *geotagging* kerusakan jalan terhapus pada tampilan dan Firebase RDB termasuk data redundansi pada *node* lain	Sistem dapat menghilangkan tampilan data yang dihapus pada aplikasi dan juga menghapus entri data pada Firebase RDB	Valid

Tabel 6.9 Hasil pengujian validasi (lanjutan)

TC-M06	Aplikasi menampilkan pesan konfirmasi bahwa laporan *geotagging* kerusakan jalan telah berhasil/batal diisukan untuk diperbaiki dan data tersimpan pada Firebase RDB	Sistem dapat menampilkan pesan konfirmasi pengisuan jalan diperbaiki dan data isu dapat disimpan/dihapus pada Firebase RDB	Valid
TC-M07	Aplikasi dapat menampilkan dialog konfirmasi dan kotak isian alasan penyalahgunaan, kemudian data isu penyalahgunaan dikirim dan tersimpan pada Firebase RDB untuk ditampilkan pada aplikasi *web* admin	Sistem dapat menampilkan dialog konfirmasi dan isian alasan penyalahgunaan, kemudian data isu penyalahgunaan tersimpan pada Firebase RDB	Valid
TC-M08	Ikon bintang terisi warna kuning untuk laporan yang telah di*vote* oleh pengguna. Untuk aplikasi yang belum/batal di*vote*, aplikasi menampilkan ikon bintang yang tidak terisi warna. Kemudian data vote dapat tersimpan pada Firebase RDB	Sistem dapat mengubah ikon bintang sesuai dengan kondisi *vote* pada aplikasi pengguna dan data dapat tersimpan pada Firebase RDB	Valid

6.1.2 Pengujian *usability*

Berikutnya dilakukan pengujian *usability* untuk menguji tingkat kepuasan dan kemudahan penggunaan, kemudahan dalam mempelajari, dan tingkat kegunaan aplikasi yang telah dirancang dan dibangun. Pengguna dalam pengujian *usability* dalam kasus ini ditujukan untuk masyarakat umum. Pengujian *usability* dilakukan menggunakan kuisioner kepada calon pengguna yakni masyarakat umum dengan kuisioner yang berjumlah 20 untuk indikator *usability* yang lebih baik (Sauro, 2013). Kuisioner diisikan berdasarkan penggunaan aplikasi *mobile geotagging* laporan kerusakan jalan pada perangkat Android secara langsung. Hasil rekapitulasi isian kuisioner ditunjukkan pada Tabel 6.10.

Tabel 6.10 Rekapitulasi kuisioner pengujian *usability*

No	Pernyataan	Jawaban					Total
		STS	TS	N	S	SS	
		1	2	3	4	5	
Usefullness							
1	Aplikasi memudahkan saya untuk melaporkan kerusakan jalan.	0	0	0	12	8	20
2	Aplikasi memudahkan saya untuk melihat jalan yang dilaporkan rusak.	0	0	3	5	12	20
3	Aplikasi sangat berguna bagi saya.	0	0	0	7	13	20
4	Aplikasi memberikan saya gambaran di mana lokasi jalan yang dilaporkan rusak.	0	0	0	8	12	20
5	Aplikasi ini membantu saya untuk mengetahui jalan mana yang telah diperbaiki.	0	0	0	10	10	20
6	Aplikasi membantu saya untuk mengetahui bagaimana gambar jalan yang dilaporkan rusak.	0	0	0	5	15	20
7	Aplikasi ini sesuai dengan apa yang saya harapkan.	0	0	1	9	10	20
Easy to Learn							
8	Saya dapat mempelajari penggunaan aplikasi ini dengan cepat.	0	2	3	8	7	20
9	Cara penggunaan aplikasi ini dapat dengan mudah saya ingat.	0	0	2	4	14	20
10	Aplikasi ini mudah untuk saya gunakan.	0	0	1	8	11	20
11	Tombol dan menu pada aplikasi ini mudah untuk dipelajari.	0	0	3	8	9	20
Easy to Use							
12	Aplikasi memiliki tampilan yang sederhana.	0	0	0	12	8	20
13	Saya dapat dengan mudah membuat laporan kerusakan baru.	0	0	7	8	5	20
14	Saya dapat dengan mudah mengisukan jalan yang telah diperbaiki.	0	0	0	3	17	20

Tabel 6.10 Rekapitulasi kuisioner pengujian *usability* (lanjutan)

No	Pernyataan	1	2	3	4	5	Total
15	Saya dapat menggunakan aplikasi ini tanpa instruksi tertulis.	0	0	5	9	6	20
16	Aplikasi praktis untuk digunakan kapan saja.	0	0	0	9	11	20
17	Warna dan tampilan aplikasi ini konsisten.	0	0	1	7	12	20
18	Saya dapat dengan cepat dan mudah memperbaiki kesalahan saya dalam menggunakan aplikasi.	0	0	3	10	7	20
Satisfaction							
19	Saya puas dengan aplikasi ini.	0	0	3	12	5	20
20	Aplikasi ini bekerja sesuai dengan harapan saya.	0	0	5	8	7	20
21	Aplikasi ini nyaman untuk digunakan	0	0	2	7	11	20
22	Saya akan merekomendasikan aplikasi ini kepada orang lain.	0	0	0	2	18	20
23	Saya ingin memiliki aplikasi ini.	0	0	0	4	16	20

6.2 Analisis

Dari setiap hasil pengujian validasi dan pengujian usability akan dilakukan analisis. Tahap analisis dilakukan untuk memudahkan mendapatkan kesimpulan dari penelitian ini.

6.2.1 Analisis hasil pengujian validasi

Analisis terhadap pengujian validasi dilakukan dengan membandingkan tingkat kesesuaian hasil yang diharapkan pada rancangan aplikasi dengan hasil uji yang diujikan pada aplikasi *mobile geotagging* laporan kerusakan jalan yang telah dibangun. Kebutuhan fungsional aplikasi dianggap memenuhi jika hasil pengujian validasi sesuai dengan hasil yang diharapkan pada rancangan. Sebaliknya, jika hasil uji dengan hasil yang diharapkan pada rancangan tidak sesuai, maka kebutuhan fungsional dianggap tidak memenuhi. Dari hasil pengujian validasi didapatkan hasil kebutuhan fungsional sistem aplikasi terpenuhi dan memiliki tingkat validitas 100%, sehingga dapat disimpulkan bahwa implementasi pengembangan aplikasi *mobile geotagging* laporan kerusakan jalan berbasis laporan sosial telah terpenuhi.

6.2.2 Analisis hasil pengujian *usability*

Dari hasil pengujian *usability* dengan menggunakan kuisioner yang diperoleh dari 20 responden, didapatkan hasil dalam bentuk akumulasi poin dari masing-masing kriteria pengujian *usability* yang ditunjukkan pada Tabel 6.10. Dalam kuisioner yang dibagikan, nilai pengujian direpresentasikan dalam bentuk angka 1 sampai 5, di mana angka 1 berarti sangat tidak setuju, angka 2 tidak setuju, angka 3 netral, angka 4 setuju, dan angka 5 berarti sangat setuju.

Indeks pengujian *usability* dapat dinyatakan terpenuhi jika nilai dari setiap aspek pengujian melebihi 60%. Untuk mendapatkan indeks presentase pengujian *usability* pada penelitian ini, digunakan intepretasi skor Likert yang ditunjukkan pada Tabel 6.11. Setelah itu dilakukan perhitungan dengan menggunakan rumus persamaan Likert. Hasil perhitungan indeks presentasi pengujian *usability* dapat pada Tabel 6.12, sedangkan hasil status dapat dilihat pada Tabel 6.13.

Tabel 6.11 Intepretasi skor Likert

Skor Likert	Interpretasi skor (interval = 20)	Pilihan
1	0% - 19,99%	Sangat tidak setuju
2	20% - 39,99%	Tidak setuju
3	40% - 59,99%	Netral
4	60% - 79,99%	Setuju
5	80% - 100%	Sangat setuju

Keterangan :
Interval 20 didapat dari pembagian 100 dengan jumlah kategori nilai skor Likert yang berjumlah 5.

Tabel 6.12 Hasil perhitungan pengujian *usability*

NO	Pernyataan	Jawaban					Total Skor	Indeks (%)
		STS	TS	N	S	SS		
		1	2	3	4	5		
Usefullness								
1	Aplikasi memudahkan saya untuk melaporkan kerusakan jalan.	0	0	0	12	8	88	88
2	Aplikasi memudahkan saya untuk melihat jalan yang dilaporkan rusak.	0	0	3	5	12	89	89
3	Aplikasi sangat berguna bagi saya.	0	0	0	7	13	93	93
4	Aplikasi memberikan saya gambaran di mana lokasi jalan yang dilaporkan rusak.	0	0	0	8	12	92	92

Tabel 6.12 Hasil perhitungan pengujian *usability* (lanjutan)

No	Pernyataan	1	2	3	4	5	Total Skor	Indeks (%)
5	Aplikasi ini membantu saya untuk mengetahui jalan mana yang telah diperbaiki.	0	0	0	10	10	90	90
6	Aplikasi membantu saya untuk mengetahui bagaimana gambar jalan yang dilaporkan rusak.	0	0	0	5	15	95	95
7	Aplikasi ini sesuai dengan apa yang saya harapkan.	0	0	1	9	10	89	89
Easy to Learn								
8	Saya dapat mempelajari penggunaan aplikasi ini dengan cepat.	0	2	3	8	7	80	80
9	Cara penggunaan aplikasi ini dapat dengan mudah saya ingat.	0	0	2	4	14	92	92
10	Aplikasi ini mudah untuk saya gunakan.	0	0	1	8	11	90	90
11	Tombol dan menu pada aplikasi ini mudah untuk dipelajari.	0	0	3	8	9	86	86
Easy to Use								
12	Aplikasi memiliki tampilan yang sederhana.	0	0	0	12	8	88	88
13	Saya dapat dengan mudah membuat laporan kerusakan baru.	0	0	7	8	5	78	78
14	Saya dapat dengan mudah mengisukan jalan yang telah diperbaiki.	0	0	0	3	17	97	97
15	Saya dapat menggunakan aplikasi ini tanpa instruksi tertulis.	0	0	5	9	6	71	71
16	Aplikasi praktis untuk digunakan kapan saja.	0	0	0	9	11	91	91

Tabel 6.12 Hasil perhitungan pengujian *usability* (lanjutan)

No	Pernyataan	1	2	3	4	5	Total Skor	Indeks (%)
17	Warna dan tampilan aplikasi ini konsisten.	0	0	1	7	12	91	91
18	Saya dapat dengan cepat dan mudah memperbaiki kesalahan saya dalam menggunakan aplikasi.	0	0	3	10	7	84	84
Satisfaction								
19	Saya puas dengan aplikasi ini.	0	0	3	12	5	82	82
20	Aplikasi ini bekerja sesuai dengan harapan saya.	0	0	5	8	7	82	82
21	Aplikasi ini nyaman untuk digunakan	0	0	2	7	11	89	89
22	Saya akan merekomendasikan aplikasi ini kepada orang lain.	0	0	0	2	18	98	98
23	Saya ingin memiliki aplikasi ini.	0	0	0	4	16	96	96

Tabel 6.13 Status pengujian *usability*

Aspek Penilaian	Status	Rata-rata Persentase (%)
Usefulness	Sangat setuju	90,85
Ease to learn	Sangat setuju	87
Ease to use	Sangat setuju	85,71
Satisfaction	Sangat setuju	89,4
Rata-rata		**88,24**

Setelah dilakukan pengujian *usability* pada aplikasi *mobile geotagging* laporan kerusakan jalan, diperoleh nilai rata-rata indeks presentase sebesar 88,24%. Sehingga dapat dinyatakan bahwa aplikasi *mobile geotagging* laporan kerusakan jalan telah memenuhi kriteria pengujian *usability*, dapat diterima, dan memudahkan pengguna dalam melaporkan kerusakan jalan.

BAB 7 PENUTUP

7.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diambil dari hasil analisis kebutuhan, perancangan, implementasi, dan pengujian pada rancang bangun aplikasi *mobile geotagging* laporan kerusakan jalan :

1. Kebutuhan dari aplikasi *mobile geotagging* didapatkan dengan cara membuat gambaran umum bisnis proses aplikasi, kemudian menganalisis aktor-aktor yang bertindak dalam sistem, dalam hal ini pengguna sebagai masyarakat pengguna jalan dan admin sebagai pengelola data. Kemudian dari aktor-aktor yang terlibat, dilakukan analisis untuk mengetahui aksi apa saja yang dibutuhkan oleh setiap aktor dan digambarkan dalam *use case diagram*.
2. Rancangan aplikasi *mobile geotagging* laporan kerusakan jalan dilakukan dengan cara membuat perancangan sistem dalam bentuk UML (*activity diagram*, *sequence diagram*, dan *class diagram*), perancangan struktur model data, dan struktur navigasi dan antarmuka aplikasi berdasarkan analisis kebutuhan berbasis daftar kebutuhan yang telah dianalisis sebelumnya. Aplikasi dirancang menggunakan Firebase *Realtime Database* yang memiliki berbagai macam fitur untuk memudahkan penggunaan data secara *realtime* dan akses data pada saat *offline*.
3. Aplikasi *mobile geotagging* laporan kerusakan jalan dibangun dan diimplementasikan pada perangkat bergerak *smartphone* dengan sistem operasi Android menggunakan bahasa pemrograman Java sesuai dengan analisis dan perancangan sebelumnya. Untuk memudahkan dalam melihat laporan berdasarkan lokasi, maka diimplementasikan tampilan untuk melihat peta visualisasi data dalam bentuk *clustered marker* dan *heatmap*.
4. Berdasarkan hasil pengujian validasi didapatkan tingkat validasi sebesar 100% dari perbandingan hasil yang diharapkan pada rancangan dengan hasil yang diperoleh pada aplikasi yang telah dibangun. Sedangkan hasil pengujian *usability* yang dilakukan menggunakan kuisioner untuk 20 responden, dengan 23 pertanyaan pada 4 macam kriteria, didapatkan nilai indeks rata-rata sebesar 88,24%. Sehingga dapat disimpulkan aplikasi *mobile geotagging* kerusakan jalan dapat diterima dan memudahkan masyarakat dalam melaporkan kerusakan jalan.

7.2 Saran

Berikut merupakan saran yang dapat diberikan untuk pengembangan aplikasi *mobile geotagging* kerusakan jalan lebih lanjut:

1. Cakupan wilayah laporan kerusakan jalan pada penelitian ini hanya meliputi wilayah Kota dan Kabupaten Malang, sehingga dapat diperluas dengan mengembangkan struktur data alamat laporan kerusakan jalan dan batasan

wilayah pada peta perangkat bergerak sehingga aplikasi dapat digunakan pada cakupan wilayah yang lebih luas.

2. Aplikasi dapat dikembangkan pada *platform* lain, seperti iOS, sehingga aplikasi dapat lebih banyak menjangkau kalangan masayarakat pengguna perangkat bergerak.
3. Pada penelitian ini, data laporan *geotagging* hanya ditampilkan dalam bentuk urutan kronologi waktu, sehingga untuk pengembangan aplikasi selanjutnya dapat ditambahkan fitur berupa pengurutan data laporan kerusakan jalan berdasarkan lokasi terdekat, atau berdasarkan jumlah *vote* laporan terbanyak yang mewakili jumlah pengguna jalan yang melalui jalan tersebut.

DAFTAR PUSTAKA

Aelani, K., & Falahah. (2012). Pengukuran Usability Sistem Menggunakan USE Questionnaire (Studi Kasus Aplikasi Perwalian Online STIK "AMIKBANDUNG"). 1-6.

Altova Inc. (2011). *Web Services: Benefits, Challenges, and A Unique Visual Development Solution.* Beverly, MA, USA.

Andi. (2013). *Step by Step Menjadi Programmer Android.* Andi dan Wahana Komputer.

Atzeni, P. (2011). *REST web service with Symfony, S.S.131.* Sestu, Cagliari, Italy, Cagliary, Italy.

Denso Wave. (t.thn.). *Geo Tagged QR Codes*. Dipetik Oktober 3, 2015, dari https://qrd.by/tracking-qr-code-html5-geolocation

Google Developers. (2017). *Firebase Realtime Database.* Diambil kembali dari Firebase: https://firebase.google.com/docs/database/

Hardiyanto, S. (2018, 02 17). *Jalan Berlubang Masih Jadi Masalah di Malang*. Dipetik 08 2018, dari www.jawapos.com: https://www.jawapos.com/jpg-today/17/02/2018/jalan-berlubang-masih-jadi-masalah-di-malang

Holzer, A., & Ondrus, J. (2011). *Mobile Application Market: A Mobile Network Operators' Perspective in Exploring the Grand Challenges for Next Generation E-Business, ser. Lecture Notes in Business Information Processing, ser. Lecture Notes in Business Information Processing* (Vol. 52). Springer Berlin Heidelberg, Germany.

Hughes, P. (2010). *The Social Report*. Bowenn Statt, Wellington, New Zealand.

IOActive Inc. (2011). *Traffic Analysis on Google Maps with Gmaps-Trafficker.* Seattle, WA 98104.

Jesdanun, A. (2012). *GPS Adds Dimension to Online Photos.* Toronto, Canada.

JSON Team. (2015). *Pengenalan JSON*. Diambil kembali dari JSON.org: http://json.org/json-id.html

Kabutz, H. (2011). *Once Upon an Oak*. Diambil kembali dari Aritma: http://www.aritma.com/weblogs/viewpost.jsp?thread=7555

Martin, F. (2010). *UML Distilled.* Yogyakarta, Indonesia: ANDI.

Nagara, C. (2009). *Analisis Hubungan Kelembagaan Antar Institusi Teknis Pengelola Jalan.* Jakarta: Universitas Indonesia.

Nielsen, J. (2012). *Usability 101: Introduction*. Dipetik 12 10, 2015, dari http://www.useit.com/alertbox/20030825.html

NSA. (2011). *Guidelines for Implementation of REST.* Savage Rd Suite 6704 Ft. Meade, USA.

Pressman, R. (2001). *Software Engineering A Practitioner's Approach* (7 ed.). Thomas Casson.

Risnita. (2014). Pengembangan Skala Model Likert. *Pengembangan Skala Model Likert, 3*, 1-14.

Rosenblatt, S. (2014). *Court Sides with Oracle over Android in Java Patent Appeal*. Diambil kembali dari CNET: http://www.cnet.com/news/court-sides-with-oracle-over-android-in-java-patent-appeal

Safaat, N. (2012). *Android Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android.* Bandung, Indonesia: Informatika.

Sauro, J. (2013). *10 Things to Know About The System Usability Scale (SUS)*. Dipetik 12 2, 2017, dari https://measuringu.com/10-things-sus/

Siegler, M. (2012). *Analyst: There's A Great Future in iPhone Apps*. Diambil kembali dari http://venturebeat.com/2008/06/11/analyst-theres-a-great-future-in-iphone-apps

Simarmata, J. (2010). *Rekayasa Perangkat Lunak.* Yogyakarta: ANDI.

Statcounter. (2015, Agustus 28). *Smartphone OS Worldwide by Installed Base*. Diambil kembali dari Statcounter.com: http://gs.statcounter.com/#tablet-os-ID-monthly-201506-201508-bar

Tom, P. (2013). *Smartphone Technology, Consumer Attachment and Mass Customisation.* Loughborough, United Kingdom: Loughborough University.

Valli, C. (2010). *Geotagging Where Cyberspace Comes to Your Place.* Lawley, WA, Australia : Edith Cowan University.

W3C. (2015). *Relationship to the World Wide Web and REST Architectures*. Diambil kembali dari W3.org: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211